FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Generic Agent Architecture for Cooperative Multi-Agent Games

João Marinheiro

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Henrique Lopes Cardoso

July 26, 2016

A Generic Agent Architecture for Cooperative Multi-Agent Games

João Marinheiro

Mestrado Integrado em Engenharia Informática e Computação

July 26, 2016

Abstract

The goal of this dissertation is to propose a high level generic architecture for the development of agents able to effectively play games with strong social components and a mix of competition and cooperation. Traditional techniques used in the context of games include a combination of heuristics and searching strategies like minimax, Branch & Bound as well as Monte-Carlo approaches; however, these techniques are difficult to apply to this category of games, due to the often enormous search trees and the difficulty in calculating the value of a player's position or move.

We propose a generic agent architecture that tackles the subjects of negotiation, trust and opponent modeling, simplifying the development of agents capable of playing these games effectively by introducing modules to handle these challenges in addition to a traditional strategic module. This architecture is split into four independent modules, taking inspiration from the structure of a wartime nation: the President, the Strategic Office, the Foreign Office and the Intelligence Office.

We demonstrate the application of this architecture by instantiating it using two different games – Diplomacy and Werewolves of Miller's Hollow – and testing the obtained agents in a variety of scenarios against existing agents. The results obtained show that the architecture is generic enough to be applied in a wide variety of games. Furthermore, the inclusion of modules to handle negotiation, trust reasoning and opponent modeling allows for more effective agents.

Resumo

Esta dissertação tem como objetivo propor uma arquitetura genérica de alto nível para o desenvolvimento de agentes capazes de jogar eficientemente jogos com um misto de competição e cooperação. Técnicas tradicionais utilizadas no contexto dos jogos incluem uma combinação de heurísticas com estratégias de pesquisa como o minimax, Branch & Bound assim como abordagens de Monte-Carlo. Contudo, estas técnicas são difíceis de aplicar a esta categoria de jogos, devido aos frequentemente grandes espaços de pesquisa e à dificuldade em calcular os valores das posições e movimentos dos jogadores.

Neste trabalho propomos uma arquitetura de agentes genérica que aborda os temas da negociação, confiança e modelação de oponentes, simplificando o desenvolvimento de agentes capazes de jogar estes jogos eficientemente através da inclusão de módulos para abordar estes temas, como complemento a um módulo estratégico tradicional. Esta arquitetura está dividida em quatro módulos independentes, inspirando-se na estrutura de uma nação em tempo de guerra: o Presidente, o Departamento Estratégico, o Departamento de Relações Externas e o Departamento de Inteligência.

Demonstramos as aplicações desta arquitetura instanciando-a usando dois jogos diferentes – o Diplomacy e o Werewolves of Miller's Hollow – e testando os agentes obtidos numa variedade de cenários contra agentes existentes. Os resultados obtidos mostram que a arquitetura é genérica o suficiente para ser aplicada numa grande variedade de jogos, e a inclusão de módulos que abordam a negociação, confiança e modelação de oponentes permite obter agentes mais eficazes.

Acknowledgements

I would like to thank my supervisor Henrique Lopes Cardoso, who was always ready to help me with useful advise and valuable criticism which allowed me to finish this dissertation. I would also like to thank the other teachers and professors that have accompanied me during the adventure of the last five years and who taught me many valuable lessons about computer science and software engineering and without which I wouldn't be where I am today. Dave de Jonge and André Ferreira were also extremely important for this work, as it is due to their work that a large part of this dissertation was possible, and for that they have my sincere thanks.

Finally, I'd like to especially thank my friends and my family, who have always been there for me and supported me during some of the toughest moments of my life.

João Marinheiro

"If you want to be incrementally better: Be competitive.
If you want to be exponentially better: Be cooperative."

Unknown Author

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

AI	Artificial Intelligence
MAS	Multi-Agent System
SDK	Source Development Kit
PDP	Package Deal Procedure
SP	Simultaneous Procedure
PR	President
SO	Strategy Office
FO	Foreign Office
IO	Intelligence Office

Chapter 1

Introduction

For decades one of the greatest challenges for Artificial Intelligence (AI) researchers has been trying to have computers emulate believable human behavior. One large part of such behavior is negotiation and bartering. Humans, as social beings, often negotiate and barter between themselves to try and improve their current situations or those of people they care about. Because of the often complex nature of these negotiations, which often involve multiple offers and counter offers, arguments, emotional appeals as well as concepts like trust, it has been hard to create AI that can accurately and convincingly imitate that behavior.

Since the start of AI research, games have been an important test-bed to develop new and interesting strategies and models. There has been extensive research using games like Chess [Dro95] or Go [SHM+16], however most research in this area relates to traditional adversarial games and makes use of extensive searching and game specific heuristics in order to find the best move [JO05].

One interesting category of games is that of negotiation games with a mix of competition and cooperation, where players are encouraged to barter and create or break deals between themselves in order to win the game. The work reported in the present document focuses on this category of games and proposes a high level multi-agent architecture for developing agents that can play a variety of such games. In this chapter, the context, motivations and objectives of this dissertation are introduced and explained.

1.1 Motivation

Negotiation games with a mix of cooperation and competition are a specific set of games with interesting characteristics for AI research, especially in the areas of negotiation and trust. There is a large variety of negotiation games with cooperative elements, some examples being Diplomacy, Werewolves, Quo Vadis?, Santiago and Genoa.

Important defining characteristics of these games are:

- Large focus on forming coalitions and negotiating deals in order to improve one's chances of winning.

- A need to balance cooperating with other players and competing.

These games often contain very large search spaces due to the high number of possible actions and agreements each player can make, which makes the application of traditional search techniques impractical.

Additionally, the strong social components of negotiation games as well as the need to sometimes cooperate with other players to increase one's odds of winning make it difficult to calculate the value of a player's position or that of a move, a key aspect of traditional search techniques. This is because such a calculation must take into account the current social context of the game, such as what coalitions exist and how much trust one has in each player, in order to be accurate. For example, if two players have an alliance pact in a game, it may not be wise to attack one of those players, even if he appears weaker than ourselves. A strong ally may quickly destroy whoever attacks the other.

Due to these characteristics it is possible to obtain much better results in these games if one is able to negotiate and coordinate with other players effectively. Having a good sense to quickly figure out the kinds of agreements that have been made between players and how to best exploit them or work around them is also something that allows a player to excel at negotiation games. Unfortunately, while humans are very good at negotiation and intuitively know who to trust, it is much harder for a computer to do so.

In order to develop effective and believable AIs for these sorts of games, and cope with the large size of the search spaces, new strategies that can effectively tackle the concepts of negotiation and trust need to be employed.

1.2 Research Goals

The purpose of this dissertation is the study and development of a high-level architecture for the development of agents capable of playing strategic multi-player negotiation games with a mix of competition and cooperation. The inclusion of ways to tackle the concepts of negotiation, trust and opponent modeling in the architecture should help direct the search for the optimal moves and agreements and improve the performance of agents when playing these games when compared with agents with no negotiation, trust reasoning or opponent modeling capabilities.

This work attempts to answer three main research questions:

- What are the main requirements to create an architecture able to effectively tackle negotiation games with a mix of competition and cooperation?
- Does the inclusion of negotiation, opponent modeling and trust reasoning strategies allow for better results in cases where the search space is too large for traditional extensive search techniques?
- Is the development of effective AI agents in multiple environments facilitated by the integration of negotiation and trust reasoning in a generic multi-agent architecture?

To work on these research questions the main objective of this dissertation was the development of a generic and expansible component based multi-agent architecture (MAS). This architecture would need to be adaptable enough to be applied to several different games and environments, allowing for the creation of agents able to play different negotiation games by simply switching out its components. In order to test the effectiveness of this architecture and as a proof of concept, we have applied it using two different negotiation games: Diplomacy and Werewolves of Millers Hollow. However, the architecture is adaptable enough to cope with a large variety of other negotiation games, and the games chosen serve only as a starting point for the work developed.

In order to successfully test and answer the research questions, the developed architecture includes integrated support for negotiation strategies, trust reasoning and opponent modeling. By using the components in the multi-agent architecture, a variety of negotiation and modeling strategies can be created and included in the different modules to easily create many different agents.

1.3 Document Structure

This document is divided into 7 chapters. The first and current chapter, Chapter 1, presents a short introduction to negotiation games, the characteristics that make them an interesting case study and the objectives for this dissertation. Chapter 2 provides an overview of different categories of games, including the category of negotiation games with cooperative elements. It also provides a brief explanation of the rules for two well-known negotiation games that were chosen for testing the proposed architecture, as well as showing some of their differences and similarities. In Chapter 3 the study of existing related work in the areas of game theory, negotiation, trust reasoning, opponent modeling and general multi-agent systems is presented. Chapter 4 introduces the proposed high-level architecture for a modular and adaptable framework for the development of agents capable of playing cooperative negotiation games, using negotiation, trust reasoning and opponent modeling. Chapter 5 describes how the proposed architecture was applied to two very different environments, those of the two chosen negotiation games. It provides a detailed explanation of the modules and the algorithms for three developed agents. The results and findings of the experiments and tests conducted using the developed agents are shown in Chapter 6. Finally, Chapter 7 presents the conclusions for the work developed during this dissertation, as well as suggestions for possible ways to continue this work.

Chapter 2

Background

In this chapter an overview of games as an interesting area of research for AI is provided as well as a description of some of the several existing game categories in the field of game theory. The characteristics and specific challenges on one such category, the category of cooperative games, will also be described. Afterwards we explain the rules for the two games, in that category, selected for the initial instantiation of the architecture described in Chapter 4. Finally, the differences between those two games as well as the specific challenges and the reasons for selecting them are also discussed.

2.1 Game Types

Games have been an area of interest in the field of AI and multi-agent systems for a long time because they allow researchers to test and explore concepts and strategies in a controlled environment. These concepts and strategies often have important applications in other areas. Examples of work in this area include the famous DeepBlue AI for chess [New02] and the more recent AlphaGo AI for the game of Go [SHM$^+$16].

Game theory studies a variety of different types of games which can be classified according to several characteristics and placed into several categories. Many different categories of games have been studied over the last decades, such as perfect and imperfect information games, deterministic and non-deterministic games or zero-sum games. And there is a large body of work in these areas with many well known strategies developed to tackle these games.

In this section we will describe some of these categories, including the category of cooperative games, in which this work is focused on.

2.1.1 Perfect and Imperfect Information Games

An important distinction to make when categorizing games is what information each player has available to him when making a decision. A perfect information game is a game where each player knows the moves every other player made before him and has full knowledge of the game state

when making a decision. Imperfect information games are games in which players may not know which moves other players have made before acting or have full knowledge of the game state.

Diplomacy and Werewolves of Millers Hollow are both imperfect information games (in the former players execute their moves simultaneously and in the latter the actions of certain roles during the night phase are kept secret). This lack of information makes the use of search strategies more difficult as explained in the work by Ritchie [Rit03].

Imperfect information games can be represented in extensive form by using information sets. Nodes in the same information set are indistinguishable from each other and the following players do not know which node in the set was picked.

2.1.2 Zero-sum and Non-zero-sum Games

Zero-sum games are games in which a gain in utility for one player represents an equal loss of utility for the remaining players. Poker is an example of a zero-sum game since the amount a player wins is exactly the same as the sum of the losses of every other player. Diplomacy is an interesting example of a game that is essentially zero-sum apart from the early stages of the game, where a player can capture neutral supply centers, thus increasing his utility without necessarily decreasing another player's utility.

2.1.3 Deterministic and Non-deterministic Games

One of the most important characteristics of a game is whether it is deterministic or not. Deterministic games are games in which the resulting state after a player's actions is only determined by the actions each player in the game chose to take and never by any stochastic event such as a dice roll. In non-deterministic games, there are states where none of the players make a choice, and where the result is determined by some form of random decision.

Examples of deterministic games include many classic board games such as Chess, Go and Checkers. Examples of non deterministic games include examples such as Risk, Settlers Of Catan as well as many computer games like Civilization.

2.1.4 Cooperative and Non-Cooperative Games

There are several proposed definitions for what constitutes a cooperative game. Since this work deals with games which contain cooperative elements it makes sense to discuss what a cooperative game is, and why it is different from traditional games like Chess.

In the field of game theory, a game is considered a cooperative game if players are able to form binding commitments with each-other. Games in which players can not create binding agreements, and thus while cooperation may happen it must be self enforcing, are usually considered non-cooperative games. It is usually assumed that communication between players is allowed in cooperative games but not in non-cooperative games [Nas51]; however this assumption has been criticized by some [Har74]. Some games cannot be neatly classified as either cooperative or non-cooperative according to these definitions, being classified as hybrid games. Diplomacy is an

example of such a game since all agreements are non-binding but communication between players is the prevalent means to achieve cooperation.

For the purposes of this work we will use a different, more general, definition of cooperative games – games in which cooperation between players is possible and encouraged but which do not necessarily have binding agreements. More specifically, we will focus on cooperative games with a *mix of cooperation and competition*.

In this type of games, sometimes also called *negotiation games*, the players are allowed to negotiate among themselves and create agreements during the game. Even though players compete among themselves, they must balance that competition with possible cooperation with other players in order to obtain the best results. Other common characteristics include the possible existence of non-binding agreements and very often the existence of very large search spaces. There are many examples of games in this category, such as Settlers of Catan and Genoa.

There are many traditional strategies used in the field of AI when dealing with games where negotiation is not present or not a very significant element. These include searching strategies like Branch & Bound or Dijkstra's algorithm as well as Monte-Carlo approaches such as Monte-Carlo Sampling. These are often complemented using many different kinds of heuristics usually dependent on the specific game being studied. However due to the characteristics of negotiation games and cooperative games these strategies are often difficult to apply with satisfying results.

The often large search spaces in these games means that using exhaustive search strategies quickly becomes computationally unfeasible. Additionally, as was detailed in Section 1.1, the strong social components of these games make it difficult to evaluate the strength of a player's position since this depends not only on the board state but also on the current social context of the game, such as alliances and agreements between players. Finally, the fact that some collaboration between players is often required to obtain good results and that some agreements can be non-binding also means that exploring the concepts of negotiation, trust reasoning and opponent modeling are essential in order to obtain efficient AIs for these games. The concepts of negotiation and trust are important not just in cooperative games but also in the real world; consequently, strategies developed to tackle these games can have other important applications.

Multi-agent systems using several different negotiation strategies and trust models have been used in this area in the past to obtain good players for some of these games, such as in the work by André Ferreira [dCF14]. In this work we will start by focusing on two negotiation games with a mix of cooperation and competition, Diplomacy and Werewolves of Miller's Hollow.

2.2 Selected Games

In order to better validate the results of this work, we decided to select two cooperative negotiation games for the instantiation of the architecture proposed in chapter 4 in player agents. The games selected were Diplomacy and Werewolves of Miller's Hollow. In this section we provide a brief description of the rules for these two games and some reasons why these games were selected. We

also compare the characteristics of these two games, and show where they are similar and where they are different.

2.2.1 Diplomacy

The game Diplomacy is one of the most well known examples of a competitive negotiation game with cooperative elements. The game is a 7 player strategy game created by Allan B. Calhamer in 1951. It is set in Europe in the years preceding World War I and each player leads one of the games' nations and must try to conquer as much territory as possible to win the game [Hil].

The map can be seen in Figure 2.1 and is split into territories that can either be sea, shore or land and may also contain supply depots. Each player begins with a set amount of territories and units depending on their nation. There are only two different types of units, which contributes to the games' ease of learning, armies and navies. They have the same power and only vary by what kinds of territories they can occupy (navies can only occupy sea and shore and armies may only occupy land and shore territories) and how they can move around the map. As the name implies, the game has a heavy focus on diplomacy and crafting alliances between players. Each turn represents one year and is split into five phases, two negotiation phases each followed by a campaign phase, and finally a reinforcement phase where every player may receive new units.

Before each movement phase, during the negotiation phase, players can talk among themselves freely. Negotiations can be both public and private and can involve any type of agreement, from alliances to sharing information. However, none of the players are forced to keep any of the agreements they make, which means trust and reputation play a large role in the game. At the end of the negotiation phase players write down the orders for every one of their units in secret, to be executed during the next phase.

During the campaign phase all of the orders written down by players in the previous phase are executed simultaneously. This can lead to devastating betrayals as each player is only sure that another player has kept his promises after his own units have moved. The orders the players can give to their units are simple, again contributing to the ease of learning of the game. Players can order a unit to hold their current position, move to some territory or provide support to some other unit (including one controlled by another player). There is also a special type of movement called a 'convoy' that for simplicity reasons will not be explained here. Whenever two units end up occupying the same territory, either because one of them moved into an occupied territory or both moved to the same territory at the same time, a standoff occurs. Standoffs are decided by comparing the strength of each unit. The strength of a unit in a standoff is the sum of its own strength plus that of any units supporting their move. Being that each unit has the same base strength value, the only way to increase the strength of a unit is through support actions. Whoever has the most strength wins the standoff and occupies the territory while the remaining units return to their original territories. If both units have the same strength, both units return to their original territories.

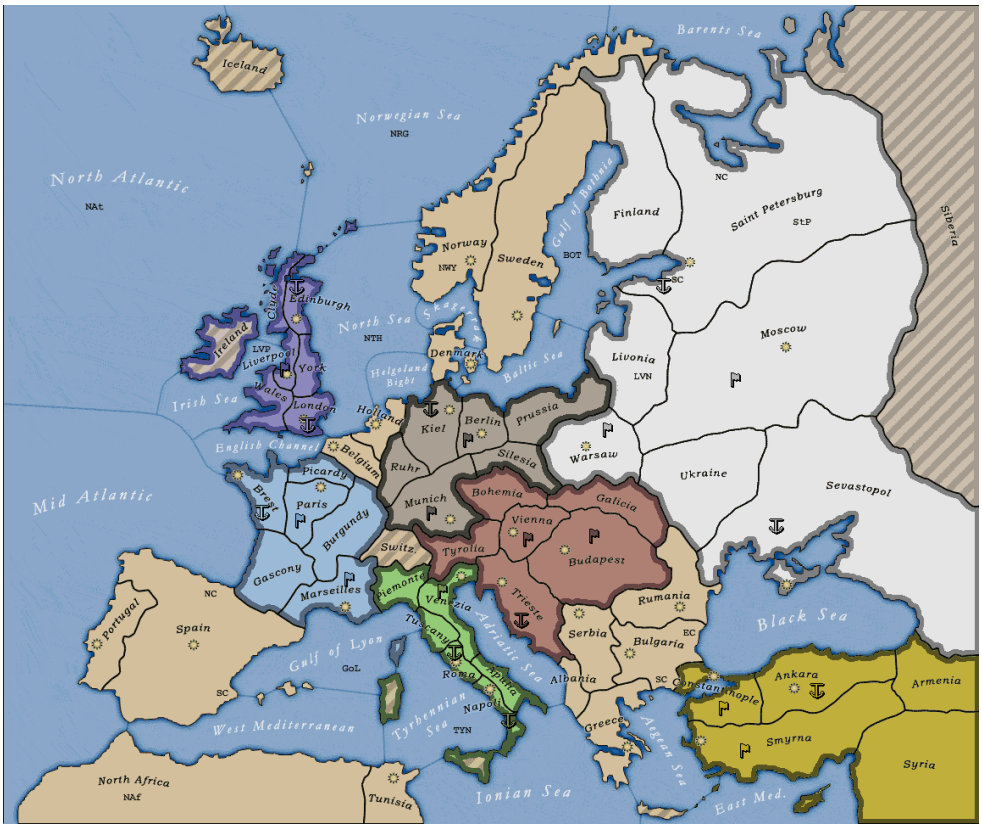

Figure 2.1: Map of Diplomacy in its initial state at the start of the game

At the end of each turn, after both negotiation and movement phases, all players may receive units in accordance with the amount of supply depots each player controls and place those units in the territories they control that contain those supply depots.

A more in-depth description of the rules for Diplomacy can be found on the official rule book published by Avalon Hill [Cal].

Diplomacy is a well researched topic in the area of multi-agent systems for a variety of reasons. Because of the variety of combinations of moves available it has a huge search space, which makes the application of traditional search techniques impractical. Being a negotiation game it has a large focus on negotiation, but the fact that it is also a competitive game where players must sometimes cooperate requires the study of notions such as trust and reputation among players. The simplicity of its rules and the fact that it is a deterministic game also makes implementing a basic game playing agent a relatively painless affair. Finally, since Diplomacy has been studied in the field of MAS for a long time, there are several frameworks available that facilitate the development of agents for this game, such as DAIDE [DAI] and DipGame [FS09].

For these reasons Diplomacy was chosen as one of the games where the proposed architecture was tested.

2.2.2 Werewolves of Miller's Hollow

Negotiation games need not even involve almost any other mechanics at all, besides negotiation. One example of such a game is Werewolves. This game is played without any board and revolves entirely around negotiation and trust. In this game players secretly take the roles of either villagers or werewolves and must try to survive until the opposing team is dead. The werewolf players know the identity of other werewolf players and must work together to kill the villagers [Asm].

The game is split into two phases, a daytime phase and a nighttime phase. During the daytime phase players discuss on who they believe the werewolves to be and why. In this sense, the negotiation consists of assurances and exchanges of information, which can either be true or false. At the end of this phase players may vote on someone whom they believe to be a werewolf to be killed, revealing their true role.

After the daytime phase comes the nighttime phase, where the werewolves can secretly choose one of the villagers to be killed. Additionally, during this phase specialist villager roles can activate their special abilities in order to try and glean any information about the identities of the werewolves, or kill or save other players. These specialist roles depend on which version of the game is being played but a common role is that of the Seer, that can see the true role of one player each night.

Since very few players have any true knowledge of other players' roles, sharing information and working together to identify the werewolves is crucial as a villager. On the other hand, the werewolves must try to remain hidden and obstruct the villagers' investigations by lying and taking out the most important specialist roles as soon as possible.

Knowing who to trust and who to accuse is one of the main challenges of this game, which makes this a very difficult game for computers to play. This heavy focus on trust and information exchange is one of the reasons this game was chosen for testing the proposed architecture. Another reason why Werewolves was chosen is because of its lack of almost any mechanic rules compared to Diplomacy. Since Werewolves has no game pieces and no board, this makes it a very different game as compared with Diplomacy. This large difference allows for the architecture to be tested in very different environments, showing that it can be used in very different circumstances.

2.2.3 Diplomacy and Werewolves of Miller's Hollow - Comparison

Both Diplomacy and Werewolves of Miller's Hollow are negotiation games with a mix of cooperation and competition that allow players to communicate among themselves in order to reach non-binding agreements, and use their actions to support or hinder each other. Both games are also deterministic and imperfect information games that work by phases.

However, despite these similarities they are very different games with different characteristics. One of the main differences is that while Diplomacy is for the most part a zero-sum game, Werewolves of Miller's Hollow is not, with several players often losing or gaining utility with certain actions. While Diplomacy is a competitive game by nature where each player is ultimately hoping to be the one to win the game itself, which encourages eventual betrayal even among long time

Table 2.1: Comparison between Diplomacy and Werewolves of Miller's Hollow

Characteristics	Diplomacy	Werewolves of Miller's Hollow
Deterministic	Yes	Yes
Information	Imperfect	Imperfect
Zero-Sum	Yes	No
Non-binding agreements possible	Yes	Yes
Simultaneous moves	Yes	Yes
Communication	Public & Private	Mostly public
Player victory	Individual	Team

allies, Werewolves of Miller's Hollow on the other hand is a cooperative game where players are divided into teams who are encouraged to cooperate among themselves and who win or lose the game as a group. The difference to a typical team based game is that in Werewolves of Miller's Hollow players do not have complete information about who is on their team and who is on the opposing team, and must thus be cautious about who they choose to trust.

Another major difference between these two games is that unlike with Diplomacy, Werewolves of Miller's Hollow has no board to speak of. Nearly the entirety of the game is based around discussion and negotiation among the players to coordinate the use of their votes and special abilities. As such the only state information that the players are required to keep is what phase the game is currently in and who else is still alive in the game.

Finally, while Diplomacy allows for secret communication between the players if they so desire, in Werewolves of Miller's Hollow all communications, apart from communications between the werewolves at night, are public and visible to all players. As such, players must take care with what they propose and reveal, or risk revealing too much of their intentions and roles.

Table 2.1 shows a summary of the comparison of the characteristics of Diplomacy and Werewolves of Miller's Hollow.

2.3 Summary

Games have been studied in the field of AI for many years and have been categorized according to many different characteristics, from the existence of random elements to the amount of information available to the players. This work focuses on one important category of games, cooperative negotiation games, that is, games in which players are encouraged to negotiate and cooperate or coordinate in order to succeed, while oftentimes still having to compete among each other to win. This category of games presents several challenges to traditional search techniques applied to games of other categories. Two of the bigger challenges are the often huge search spaces that have to be traversed in these games and the requirement to understand the social context of the game in order to calculate the value of a position or move.

In order to validate the results of the work reported in this document we selected two cooperative negotiation games which we will use to test the proposed architecture. These games are

Diplomacy and Werewolves of Miller's Hollow, both with very different rule sets and characteristics, but also some key similarities.

In the following chapter we will review and analyze previous existing works in the areas of multi-agent systems, negotiation, trust and cooperative games.

Chapter 3

Related Work

This chapter provides an overview of existing work in the areas of game theory, negotiation, trust reasoning, opponent modelling and negotiation games, including a description of several existing bots for this kind of games. A short overview of existing frameworks and engines for negotiation games will also be provided.

3.1 Game Theory and Game Representations

When dealing with cooperation and conflict it is important to understand some important concepts of game theory. Game theory is a field of mathematics that deals with how logical and intelligent decision makers act in a variety of games and situations involving conflict and cooperation. John von Neumann was the first to explore this area with his theory regarding the existence of mixed-strategy equilibria in two-person zero-sum games [Leo10].

Game theory initially studied perfect information zero-sum games but today studies a variety of game types and categories.

Games studied in the field of game theory are well defined models that must specify the players of the game, the information available to each player at each decision point as well as the available actions and the payoffs for each outcome [Ras06]. This information can then be used to calculate a strategy or a set of strategies that when employed by every player, no player can unilaterally obtain better results by deviating from these strategies. When every player uses these strategies a Nash equilibrium is achieved since no player has any incentive to deviate and use other strategies. Calculating the Nash Equilibrium strategies for a game is a key aspect of game theory. Unfortunately, for many games it is computationally impractical to calculate such an equilibrium. In these cases approximations to a Nash equilibrium must be found instead.

Games can be represented in two forms in game theory: extensive form and normal form [Gib92]. The extensive form can be used to represent games where the order of player actions is important and is usually represented using decision trees. These trees specify when each player can move, what information this player has available and the payoff for each combination of moves. Normal form is usually used to represent games in which players act simultaneously or without

knowing the actions of the other players. A normal form representation consists of a function that associates a payoff for each player for every possible combination of actions, typically represented using a matrix.

Every extensive form game can have an equivalent normal form representation. However, for some games, the transformation from extensive form to normal form may result in a dramatic increase in the size of the representation which may make it computationally impractical [LBS08].

3.2 Negotiation Strategies

According to Beer *et al.* [BDL+99] there are three main research topics in multi-agent negotiation. These topics are the negotiation protocols, the negotiation objects and the reasoning models.

3.2.1 Negotiation Protocols

The negotiation protocols regard the rules for how the communication between the agents must be done. The protocols dictate any agents or third parties that can participate in the negotiation, what negotiation states there are and what actions can the participants take at a given time.

One of the most widely used negotiation models is Rubinstein's alternating offers model [Rub82] to which many negotiation protocols, such as the Monotonic Concession Protocol [RZ], adhere.

One aspect of these protocols is the communication languages used. Examples of widely used communication languages are KQML and FIPA ACL. For the purposes of a generic negotiation system for a variety of games, these communication languages must be powerful enough to allow agents to express a wide range of arguments and arrangements. A game like Diplomacy requires agents to be able to express anything from alliance proposals, to threats, to information exchanges. Some systems define their own communication languages that are domain specific, as is the case with the architecture defined by Kraus and Lehmann [KL95], where they defined a communication language based on the observation of real Diplomacy players.

3.2.2 Negotiation Objects

This area refers to what objects are part of the negotiation procedure, whether those objects are divisible or not and if they can be changed during negotiations or not. Objects can be anything from an alliance to a request for support in an attack to a trade for a specific good. One aspect of the negotiation is finding what objects should be negotiated about, and a negotiation can involve only one or several objects simultaneously. When a negotiation involves more than one object it is referred to as a multi-issue negotiation. Many negotiation games involve multi-issue negotiation and so the agents created must be able to efficiently negotiate deals that involve more than one item at a time.

There are two types of bargaining frameworks to deal with multi-issue negotiation: issue-by-issue negotiation and simultaneous negotiation [Sah06]. In issue-by-issue negotiation, objects

are first ordered and then negotiated sequentially. In simultaneous negotiation all objects are negotiated simultaneously. One way to do this is using the package deal procedure (PDP), where each proposal includes an offer for every item being negotiated, and offers may only be accepted or rejected as a complete package [FWJ06]. PDP can be quite computationally intensive when a deal involves a large number of objects, and may lead to drawn out negotiations since deals may only be rejected or accepted as a whole. Another option is the simultaneous procedure (SP) method, where each item is negotiated independently but in parallel.

One important factor considered in multi-issue negotiation is the calculation for the utility of the deal, which may be non-linear. That is to say that some combinations of objects may have more or less value than their isolated utility when negotiated together. For example when bartering, negotiating a car and fuel together has a higher utility value than negotiating for each item separately. In these cases PDP can be more useful than SP, because items are negotiated as a package and not independently.

3.2.3 Reasoning Models

Reasoning models describe how the agents participating in the negotiations decide what actions to take. This is dependent on the chosen negotiation protocol and on the objects being negotiated. There are several known negotiation strategies that help an agent decide how it should proceed. Examples of these strategies that determine when an agent concedes and how much it concedes are the conceder strategy [Pru81], the Boulware strategy [Rai82], Tit-For-Tat strategy and the Zeuthen strategy [RZ].

The conceder strategy makes large concessions at the start of the negotiation and progressively smaller concessions as the negotiation approaches the deadline. The Boulware strategy does the reverse, making small concessions in the beginning and larger concessions as the deadline approaches. The Tit-For-Tat strategy uses information about the other party's previous concessions to guide an agent's actions. The Zeuthen strategy is based on the willingness to risk conflict of each agent, and states that the agent that is less willing to risk conflict should concede first, and should concede just enough so that one of the other agents will have to concede on the following round of negotiation. These strategies can also be combined to create more efficient strategies depending on the situation [FSJ98].

Matos *et al.* studies the relative performances of these negotiation strategies and provides a basis for deciding which strategy to use when the other party's strategies are known [MSJ98]. Other approaches build on this work to deal with situations in which agents do not have complete information about each others strategies, for example by using non-linear regression analysis to predict the opponent's strategies based on his previous offers [Hou04].

Frequently however, agents do not have an easy way to know the utility value of an offer for other agents which may lead to dropped negotiations and sub-optimal deals. This is the case in many competitive negotiation games, where players may not want to reveal that information to increase their own gains. As such, alternative strategies to cope with this uncertainty have been

proposed, such as the use of a mediator that has access to the agents' private information and proposes deals with that information in mind or the use of opponent modeling strategies.

One interesting approach in this area is the NB^3 algorithm proposed by Dave de Jonge and Carles Sierra [Sie11, dJ13]. This algorithm is based on the Branch & Bound search algorithm and mixes negotiation with search so that the former can direct the latter. It was designed to be used in environments where the search space is too large for traditional search techniques to be employed, and where better solutions often require cooperation among several agents. The algorithm was tested using a version of the Traveling Salesman Problem called the Negotiation Traveling Salesman Problem where several salesmen must coordinate among themselves which cities to visit in order to minimize their own travel costs.

NB^3 attempts to expand the nodes, representing partial plans, that not only decrease the cost for the agent itself, but also for other agents that might be willing to accept that plan. This is because if an agent takes too long to convince other agents to accept its plans, other agents may enter into contradicting agreements that force him to follow the initial, more costly plan. The results from previous negotiations also allow plans that go against the accepted agreements to be pruned from the search tree.

While this algorithm was modeled as a minimization problem, it may easily be adapted to be used with maximization problems. Diplomacy is proposed as one of the possible applications for NB^3 since it has many of the same characteristics as the Negotiation Traveling Salesman Problem. However, this approach assumes that agents are honest and keep their agreements after they've been made.

3.3 Trust Reasoning and Opponent Modeling

Negotiation games revolve around social interactions between players. These interactions are usually in the form of agreements, contracts and information exchanges between players. However, when these contracts are non-binding, relying on each player's will to abide by them, there is a certain risk when accepting a contract that it will be broken by one of the parties. For this reason, in order to protect themselves players need some way to calculate how likely a contract is to be kept by each party. That is, how much does the agent trust that some outcome will happen. This is not only reliant on the integrity of a particular agent, that is, whether it tries to the best of its ability to fulfill his part of the agreement, but also on his ability to obtain the outcomes desired.

In order to get an accurate measure of trust, several factors need to be taken into account, such as a player's integrity, his ability, past results of deals with that player, information from other trustworthy players about that player and the goals that player might have at the time. For this it is also extremely important to have an accurate opponent model, so as to better understand the opponent's preferences and goals. These can help contextualize the trust on that player. For example, it may not be wise to trust a player to keep a deal that we know goes against its current goals, regardless of how trustworthy that player has been so far.

Trust reasoning and opponent models are important concepts in negotiation games because they allow agents to measure how risky it is to enter into an agreement as well as decide what the best agreements to propose are based on what the opponent is likely to accept. It is important then to study how to accurately model these concepts in a multi-agent system.

In general there are two ways that an agent can obtain information about the trustworthiness of another agent. It can obtain information directly from his interactions with the agent or it can ask his peers for information about the agent. The second method is connected to the concept of reputation, which represents what other agents think about someone.

Marsh [Mar94] was the first to model trust computationally, taking inspiration from sociology and economics. He proposes a model where trust varies between 1 and -1 and agents trust each other on specific situations, depending on the risk associated with that situation as well as the importance given to it by the agent. One disadvantage of this approach is that the trust value is not altered significantly when an agent defects, allowing it to continue exploiting its opponent on the following interactions.

Burnett [Bur11] proposes a categorization system for quickly calculating a trust value for newcomer agents in environments where agents frequently join and leave the global population. This is done by stereotyping the agents by their observable features. Stereotypes are represented as decision trees that are traversed using the perceived features of an individual, returning a predicted trust value that can quickly be used.

More recently in her work Joana Urbano presents the SOLUM model [Urb13], which evaluates agents on their ability, integrity and benevolence. In this context ability represents the general competence of an agent at a certain task, integrity represents how honest an agent is when attempting to fulfill his obligations and benevolence represents a feeling of goodwill between trustee and the truster. SOLUM also extracts context information from patterns in past interactions between agents, allowing for a more accurate prediction of trustworthiness.

In the field of cooperative games there have also been some attempts to use trust reasoning to improve the effectiveness of agents when there is the possibility for the creation of non-binding agreements. DipBlue, proposed by André Ferreira [dCF14], uses a simple trust reasoning method based on a trust ratio and a friction ratio. In this model the amount by which the trust in an agent changes is both based on the actions of the agent, as well as the current disposition towards it and its current trustworthiness. If an opponent takes a hostile action against DipBlue, the trust in that opponent is more (negatively) affected in cases where DipBlue had high trust in that opponent or was at peace with it, than if it had low trust or was at war.

Many other computational trust models exist, as identified in surveys such as [JIB07] and [PSM11].

Since often times it is in the best interest of agents to keep their goals, utility functions and negotiation strategies secret in order to gain the upper hand in negotiations, it can be difficult to know which deals an agent can propose to other agents that are both likely to be accepted and likely to be kept. Opponent modeling is one way to tackle this issue. Opponent modeling consists on the creation of models describing how another agent is likely to act and what its preferences are by analyzing this agent's actions [JAMSU11]. These models can contain predictions such as what

the opponent's goals in the negotiation are and what negotiation strategy it is likely to be using. These can then be used to inform the negotiation process with that agent, so that better deals can be proposed, that avoid giving up too much utility or are unlikely to accepted or kept by the other agent.

There has been some interesting work in this area in the context of cooperative games. Faratin *et al.* [FSJB99] proposes a negotiation strategy where agents propose deals that are more similar to deals previously proposed by the opponent, either by a trade-off mechanism that keeps the same value for the proposals or by removing problematic issues from the negotiation. In his work, Krimpen *et al.* [vKLH13] propose a simple method where more utility is assigned to issues that remain constant over several offers.

Perhaps one of the more interesting works in the area of competitive negotiation games with cooperative elements is the work by Afiouni and Øvrelid [AØ13] that builds upon the opponent modeling techniques described by Krimpen *et al.* In their work, they propose a negotiating agent that uses weighted constraints to evaluate offers. By watching the variation in issues in offers proposed by its opponents, this agent can add or remove constraints from its opponent model, or alter their weights. It then proposes new offers by solving a prioritized constraint satisfaction problem (PCSP).

The NB^3 algorithm proposed by Dave de Jonge and Carles Sierra [Sie11, dJ13] also approaches this concept by using the predicted utility of the proposals that other agents have made or accepted to calculate the expansion heuristic for the branch & bound search. By doing this the search is naturally directed towards nodes with a probable higher utility value for the other agents involved in the negotiation, thus finding deals that are more likely to be accepted by these players. However the authors do not go too much into this concept and leave the prediction of opponent goals and preferences as a domain specific problem. They also make no attempts to predict the negotiation strategies of other agents.

3.4 Existing Approaches to Cooperative Games

There are some existing approaches that share similarities with our work and from which we took inspiration. This section will detail some of these existing approaches. The purpose of this is not to provide an exhaustive list of game playing agents but to discuss some agents created to play cooperative negotiation games, and which deal with many of the same issues that this work deals with. Unfortunately, while there exist many agents for Diplomacy, no existing approaches have been found for the game of Werewolves of Miller's Hollow.

3.4.1 Diplomacy Playing Agents

The game of Diplomacy has been a well known test-bed for distributed agents for a long time and there have been several implementations of different agents with different performances and capabilities over the years. These agents are generally separated into two categories: no-press agents with no negotiation capabilities and negotiating agents. Some implementations are now

discussed. While some of these implementations do not involve trust reasoning or negotiation, they nevertheless provide useful insight into the architecture of a game playing agent.

One of the simplest agents implemented for Diplomacy, DumbBot was developed in two hours by David Norman [Jon10]. This bot has no negotiation capabilities and uses a simple heuristic to decide its actions by preferring to choose moves that weaken its strongest opponents. DumbBot assigns a value to each territory that depends on who controls it and what units are around it, and then it assigns actions to every unit depending on those score values.

While the method used is very simple, DumbBot obtains fairly good results and is frequently used as a benchmark for other Diplomacy agents.

Another interesting agent is DarkBlade, which was created with the objective of mixing several existing strategies together in order to obtain the most efficient no-press agent possible [Rib08]. This agent is split into several sub-agents that negotiate among themselves in order to choose the best moves. This agent successfully surpasses DumbBot in performance. Another agent, HaAI, uses a similar multi-agent structure within itself where its components, each representing a unit, negotiate among themselves in order to choose the best actions [JH05].

One of the most important and influential negotiating agents is the Israeli Diplomat. The architecture of the Israeli Diplomat was designed to be a general negotiation architecture to be applied in a variety of situations. This architecture was used to create a Diplomacy playing agent [KL95].

The Israeli Diplomat tries to mimic the structure of a war-time nation. It consists of several components working together to choose the best course of action. These components are the Prime Minister, the Ministry of Defense, the Foreign Office, the Military Headquarters, Intelligence and the Strategies Finder. The Prime Minister directs the actions of every component, and defines the general personality of the agent. The Ministry of Defense is responsible for planning and analysis, directing the work of the Foreign Office, the Headquarters and the Intelligence. The Foreign Office handles communication with other agents. The Military Headquarters are responsible for deciding the moves that the agent makes. Intelligence is responsible for gathering information on other players. In Figure 3.1 the general organization of the Israeli Diplomat architecture is shown.

The diplomat keeps a knowledge base of the relations it believes each nation has with eachother as well as any agreements it has, its intention to keep them and its trust that others will keep them. This knowledge base is updated by the different modules as the game progresses, and affects every decision that the diplomat takes.

Some of these modules are further divided into local-agents that communicate among themselves. For example, the Foreign Office contains several agents, each tasked with negotiating with a different player. These allow the Israeli Diplomat to split the problem space into manageable chunks and handle conflicting actions. This highly modular architecture allows the Israeli Diplomat to efficiently search for the best strategies and deals by independently considering several possibly conflicting courses of action.

The Bordeaux Diplomat, like the Israeli Diplomat, separates the negotiation from the strategical analysis of the game. It is composed of two modules, a negotiator module and a strategical

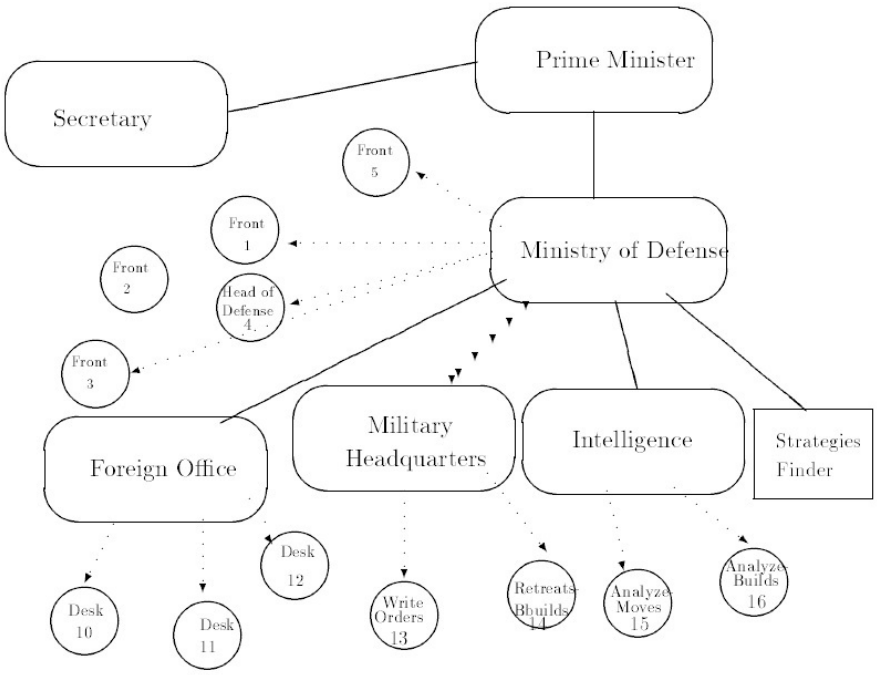

Figure 3.1: General Structure of the Israeli Diplomat, from [KL95]

core module [HL95]. However the authors focus almost exclusively on the strategic core module, offering only a vague description of the negotiation module.

The strategical core has no information about what nation it is playing or what its agreements are and is merely an impartial observer whose job it is to calculate strategies for any nation. To calculate the best strategies it uses an evolutionary algorithm that is an improved best-first search called Refined Evolutionary Search. This algorithm mutates a set of actions until it finds the best strategy according to the current constraints.

The negotiator module can query the strategical core for the best strategies to execute and other useful information. By asking the strategic core for any moves that other players are likely to make in different conditions and comparing them with the actual moves executed the agent can infer the reliability of a player, what alliances a player has made, how likely he is to betray it and whether an agreement with another player is in its best interests.

This agent maintains a friendliness matrix as a means to model the loyalty of other players. By keeping a minimum gain value for each player and calculating if that player can achieve that gain in the field, it can predict if a player is likely to betray him.

Another interesting approach in this area is that of D-Brane [dJ15], an agent developed by Dave de Jonge that makes use of the NB^3 algorithm as well as a complex strategical module to find the best sets of moves to negotiate and play. This agent uses a basic form of opponent modeling by using the utility values of deals previously proposed and accepted by its opponents as a way to direct the search for better solutions, however it does not make an attempt to explicitly

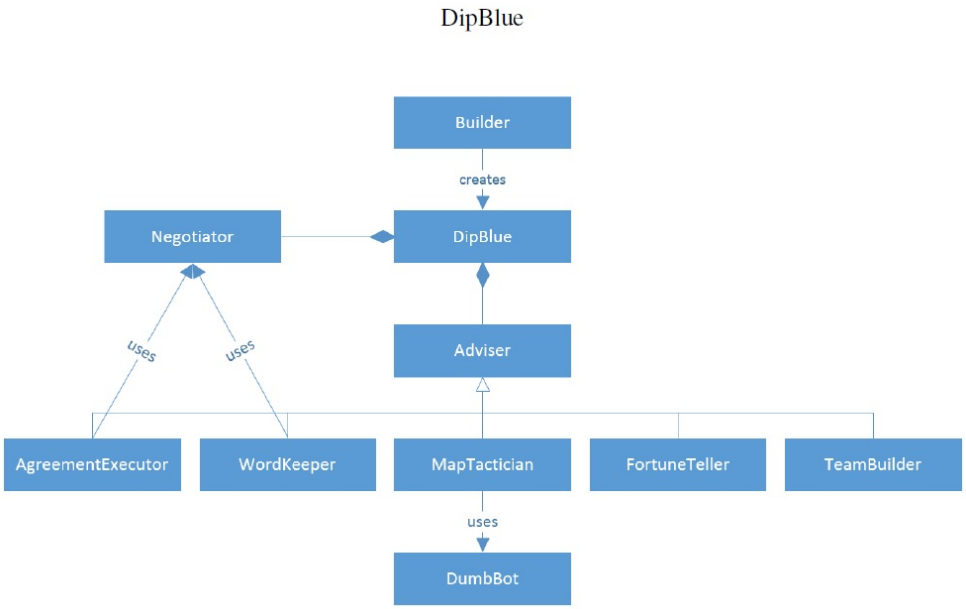

Figure 3.2: General Structure of DipBlue, from [dCF14]

predict an opponent's goals or strategies. Another aspect that this agent lacks is the ability to negotiate coalitions with other agents as well as joint moves for future phases of the game, having no negotiation strategy for these kinds of deals.

Finally, DipBlue is a negociating agent for Diplomacy inspired by the Israeli Diplomat architecture. The agent is split into several modules called Advisers, that together decide the actions that the agent takes [dCF14]. This structure is shown in Figure 3.2. Each adviser receives the evaluated move scores from previous advisers and alters them according to its role. The base adviser is inspired by DumbBot and uses the same scoring heuristic. This score is then changed by other advisers to promote support actions for the units, promote actions that keep agreements with its allies and encourage the agent to attack players that it distrusts.

In order to model the trust value of each player, DipBlue keeps a trust matrix that is updated as the game is played [FCR16]. If a player performs hostile actions against DipBlue, such as attacking it or breaking an agreement, its trust value diminishes. If a player performs friendly actions, or refrains from doing hostile actions, its trust value increases. DipBlue is more likely to accept agreements and help players with which it has a high trust value, and attack players with a low trust value.

DipBlue does not have full negotiation capabilities, lacking the ability to ask and give information or threaten players.

In Table 3.1 a summary of the different Diplomacy playing agents is presented, along with their approaches to strategic search, negotiation, trust reasoning and opponent modeling.

Table 3.1: Diplomacy playing agents

Agent	Strategy (search)	Negotiation	Opponent Modeling & Trust Reasoning
DumbBot	Province destination value heuristic.	None.	None.
Darkblade	Sub-agents negotiate among themselves to find the best moves.	None.	None.
HaAI	Sub-agents negotiate among themselves to find the best moves.	None.	None.
Israeli Diplomat	Calculates a loss and profit value for each order.	Starts negotiating general intentions and gradually decides specific actions.	Predicts player relations and trustworthiness
Bordeaux Diplomat	Refined Evolutionary Search algorithm.	Negotiates with the help of its strategical module to find good moves to propose.	Finds the likelihood of betrayal by finding the maximum gain a player can obtain in the field.
D-Brane	NB^3 algorithm and a complex heuristic.	NB^3 algorithm.	Basic opponent modeling by using the NB^3 algorithm.
DipBlue	Province destination value heuristic.	Allies against the strongest players and requests supports its allies.	Calculates a trust and friction value for each player.

3.4.2 Other Games

In the area of real-time computer games the work by Afiouni and Øvrelid [AØ13] presents an application of academic AI techniques to the field of video-games. They developed a modification for the AI system in the negotiation game Civilization IV by Firaxis Games, using the game's Source Development Kit (SDK). While they do not tackle the problems of with whom to negotiate or when to negotiate, leaving the base game's AI to make those decisions, they present a negotiation strategy based around solving a Prioritized Constraint Satisfaction Problem and modeling their opponents' preferences using both Bayesian Learning and Frequency Modeling.

3.5 Zillions of Games

An interesting approach in the area of general AIs that can play a variety of games is the Zillions of Games system developed by Jeff Mallet and Mark Leffer [Cor]. This is a system where players can define a wide variety of two dimensional abstract board games using rules files written in a specific language (ZRF files). The platform can then read these files and generate the game as well as intelligent generic AIs that can play it.

While the system is limited in the rules and layouts of the games it can generate, and its AI features no negotiation or trust reasoning capabilities, it is nevertheless an interesting example of a general AI that is adaptable enough to play in a variety of different environments.

3.6 PGDL

PGDL is a system that allows users to define several different Poker variants using a specific game description language [Cor13]. The system allows users to play the defined Poker variants against simple agents able to play any game defined using the system. Like Zillions of Games, while the system is limited to poker variants, and features no negotiation or trust reasoning capabilities, it provides another interesting example of a general approach to model, interpret and build automatic players to a family of games.

3.7 Technologies

Considering the interest and usefulness of games to test concepts in multi-agent systems, there are several frameworks that allow users to write and test their own agents.

For Diplomacy there are two important test-beds, DAIDE and DipGame. Both of these allow for users to write Diplomacy agents with negotiation capabilities using well defined communication protocols, and test them using a graphical interface. DAIDE is the oldest of the two, created by the DipAI organization and built using C/C++. It uses its own well documented communication language for handling messages between the clients and the server and communication between agents [DAI]. Being the most used Diplomacy development framework, it has a large number of resources available and many agents built using it. DipGame is a framework developed by IIIA-CSIC of Barcelona and built on top of DAIDE [FS09, FS11]. It uses its own L Language standard for communication. Being built using Java, DipGame has the advantage of being cross-platform. Because DipGame is still a relatively new platform, its documentation is not as extensive as DAIDE's and it does not have as many agents created using it.

JSettlers2 is a Java based test-bed for multi-agent negotiation using the Settlers of Catan board gam [Tho03]. Like DAIDE and DipGame, this framework also contains a graphical interface, but it is not as well documented as the Diplomacy test-beds.

Another useful tool for AI development in the context of negotiation games is the Source Development Kit for the Civilization IV game, developed by Firaxis Games. The SDK uses C/C++

and allows users to create their own AIs for the game using a set of classes [Civ]. This tool is well documented but one disadvantage it has compared to the other mentioned frameworks is that it lacks extensive testing and debug tools.

3.8 Summary

There has been extensive research in the area of negotiation in multi-agent systems in the past decades. However a large part of this research is based on perfect knowledge of other agents' preferences and negotiation strategies. In the context of negotiation games, in order to accurately model real world situations we must assume that players do not have perfect information about their opponents. Therefore some kind of opponent modeling approach, such as the one used by Afiouni and Øvrelid, is needed.

Another important concept is the concept of trust. Trust is a measure of how likely an agent expects a certain outcome, dependent on another agent, to happen. Trust reasoning is important in order to reduce the risk for agents in situations where they may suffer defection by other agents.

With regards to agent architecture the Israeli Diplomat architecture is one of the most important works in this area, providing a modular and flexible architecture that can be adapted to a number of situations. Several negotiating agents such as DipBlue take inspiration in this architecture for these reasons. Other interesting examples of approaches to generic architectures for games are the Zillions of Games software and the PGDL system.

In the following section we will describe the general proposed high level architecture, which takes inspiration from the Israeli Diplomat and DipBlue, and tackles the concepts of negotiation, trust reasoning and opponent modeling.

Chapter 4

A Generic Architecture For Cooperative Game Playing Agents

This chapter provides details about the problem and the challenges that a generic agent architecture for negotiation games with a mix of cooperation and competition should overcome, as well as introducing the proposed architecture and its different modules.

4.1 Architecture Proposal

In Section 1.1 we detailed the characteristics of negotiation games with a mix of competition and cooperation that make them an interesting case study in the field of multi-agent systems. The often large search space in these games makes the use of traditional search techniques impractical; moreover, a social dimension in such games makes calculating the strength of different positions and moves difficult. Additionally, despite being competitive games they often require a great deal of coordination and collaboration between different players in order to obtain the best outcomes. These problems encourage the use of techniques that can use negotiation, trust reasoning and take into account the social context of the game to direct the search for the optimal moves.

In order to facilitate the development of agents able to effectively play these games we propose a generic architecture that tackles these issues, allowing for the creation of agents capable of negotiation, trust reasoning and opponent modeling in a simple way. In order to be used in a variety of games and environments this architecture is required to be as generic as possible, making no assumptions about negotiation protocol, negotiation strategy, player goals, and search strategy. As such the proposed architecture, which we dubbed the ***Alpha*** architecture, is modular and based on the architecture of the Israeli Diplomat and DipBlue. Figure 4.1 shows a simplified overview of the Alpha architecture while Figure 4.2 shows a more detailed view of the modules proposed.

Like the Israeli Diplomat, it is based on the structure of a wartime nation, with four different independent modules:

- **The President:** a control module that coordinates other modules and takes the final decisions.

- **The Strategy Office:** an adviser modules that calculates and suggests good strategies to the President.
- **The Foreign Office:** the module in charge of all communication with other agents.
- **The Intelligence Office:** an adviser module that advises the president on what its opponents are likely to do.

This structure allows the architecture to have a clean separation between the different independent modules for the different subjects of negotiation, opponent modeling, the strategic and tactical evaluation of the game and the high level agent personality and overall strategy. As a result the user can easily and effortlessly swap out the modules at will and decide what capabilities the agent should have, allowing for great freedom in designing agents. Different modules can also be individually altered and tweaked without interfering with each other, allowing for a simple way to make tweaks and changes to certain capabilities of the agent. By creating a relatively small quantity of these modules and combining them in different ways the developer can create a large variety of agents with different behaviors and capabilities in a short time.

In the following sections we will explain the functions and structure of each of these modules individually.

4.2 The President

The President (PR) acts as the central module for the agent, holding its personality characteristics, coordinating the other modules, defining the overall high level strategy of the agent and being in charge of selecting and ultimately executing the moves for the player.

In order to do this the PR keeps a knowledge base of everything it needs to know about the environment and its opponents. This knowledge base is then also used and modified by the remaining three modules, allowing the PR to make the best decisions possible with up-to-date useful information regarding the environment. The information contained in the knowledge base is the following:

- The current state of the game, such as the state of the board and the current game phase.
- The moves played during the course of the game by each player.
- The current player goals for the game. Goals are game-specific and can range from what territories to capture to what players to eliminate or any other kind of goal the player might have during the course of the game.
- The lists of deals confirmed, completed and proposed by itself and other players over the course of the game. Like goals, deals are specific to each game and can be, among other things, peace proposals, requests for information or commitments to specific moves.

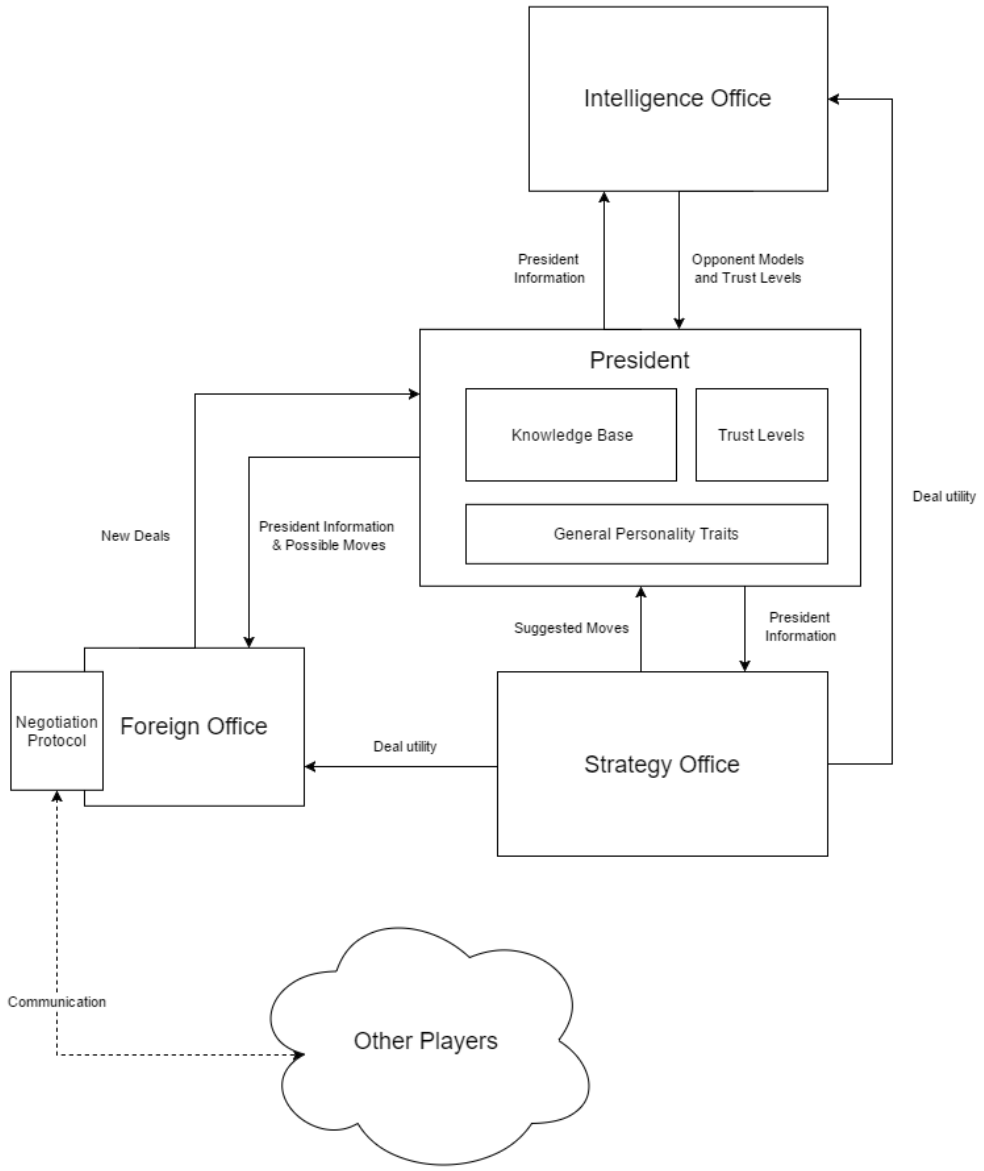

Figure 4.1: General high-level structure of the Alpha architecture

- The player's current disposition towards other players, such as who are its allies and enemies.
- The opponent models for each other player, including predicted goals and negotiation strategies.
- The general trustworthiness levels of each player.
- The trustworthiness levels of each deal. Separating these from the trustworthiness levels of the players allows for more accurate predictions, as a player may be trustworthy in general, but fail to keep agreements that go against its main goals.

Additionally, the PR also has a set of personality traits that can be defined depending on the game being played. These govern the general strategy of the PR, such as how aggressive it is, how trusting of other players it is or how prone to taking risks it is. Finally, the PR also keeps lists of moves and deals suggested by the other modules, which the PR can then choose to execute, depending on its personality traits, goals and knowledge of the environment.

When defining the PR module the developer must define what constitutes a deal, a move and a goal because these are game-specific concepts that are difficult to generalize. Different games such as Diplomacy have very different possibilities for what moves and deals a player can make compared to a game like Werewolves of Miller's Hollow. These definitions specify the game-specific elements that are stored in the knowledge base and used by all modules when making calculations.

One important role of the President is deciding what overall goals the agent is striving for and what relative importance to attribute to each goal. This information is then used to inform the behavior of other modules, such as what moves are suggested by the Strategy Office or what deals are accepted by the Foreign Office. This allows the PR to dictate the overall strategy it wants to follow to its subordinate modules, allowing them to focus on the individual details of what actions are more likely to result in attaining these goals.

Different PR modules allow the developer to customize the agent's general strategy and personality allowing for different player archetypes.

4.3 The Strategy Office

The Strategy Office (SO) has the purpose of analyzing the strategical situation of the game and suggesting good moves to the PR. It contains most of the game-specific heuristics, evaluating the utility of possible moves and deals, and as such is highly dependent and adapted to the specific game being played. In addition to that it also defines the search strategy used to explore the possibility space for different moves.

This module is generally separated into two parts: the search strategy used when looking for moves to analyze and the evaluation method used for calculating the utility value of moves and deals. While the search strategy used can generally be applied to different environments relatively easily, the tactical evaluation is, in general, entirely dependent on the game being played, as it relies on specific knowledge about the game's rules in order to calculate what constitute good moves.

In order to find the best moves, the SO has access to the PR's knowledge base, being able to make use of the information contained there to evaluate the utility of different moves and deals. The SO can for example make use of what the PR knows about the current state of the board, what deals have been confirmed and what its overall goals are to suggest a possible move that adheres to any deals that have been made and is likely to help the PR obtain its goals. When the PR requests move suggestions and the SO finds a good move it suggests it back to the PR who stores it for consideration on its internal list.

Swapping the Strategic Office allows a developer the choice among different search strategies and heuristics for the game, which can have a major impact on a player's effectiveness.

4.4 The Foreign Office

The Foreign Office (FO) has the purpose of managing any interaction with other players and negotiating deals and coalitions in a way that best allows the PR to execute the moves it is considering in order to fulfill its goals. When the PR requests the FO to negotiate with other players it sends it a list of what moves it is considering for which the FO should attempt to find supporting deals. Using this information as well as any other information available in the PR's knowledge base that the FO finds useful, this module then autonomously communicates with other players and decides what deals to propose, reject and accept.

The Negotiation Strategy used is defined in this module and determines what deals are proposed and accepted and what concessions the agent is willing to make. This module also defines the negotiation protocol used by the agent when communicating with other players. The decision of what protocol to use is often dependent on the game being played or even the specific development framework on top of which the agent is being implemented.

Like the SO, this module has access to the PR's knowledge base in order to make the best decisions. It can, for example, make use of the trust values stored in the PR's knowledge base to know which agents are more likely to keep their deals, and as such better candidates for negotiation. While the FO's purpose is to negotiate good deals and not to decide strategically what constitutes a good position for the player to be in, it must also have a way to know when a deal is good for the player and thus should be considered for proposal or be accepted, and when it is not. To do so, the FO communicates with the SO, requesting it to calculate the utility values of deals it finds or are proposed by other players. When a deal is proposed, confirmed or completed the FO updates the PR's knowledge base adding these deals to the appropriate lists.

Swapping the FO allows a developer to customize the negotiation capabilities of the agent, allowing the use of different negotiation and concession strategies, or even removing the FO altogether for an agent with no negotiation capabilities.

4.5 The Intelligence Office

The Intelligence Office (IO) is in charge of calculating the trust values and opponent models for the different players in the game. It has access to the PR's knowledge base and analyses, among other information, the past moves and proposed deals of other players in order to predict their goals, how likely they are to keep each deal they are involved in and what their general trustworthiness is. It then updates these values in the PR's knowledge base so that they can be used by the PR and the other modules to make informed decisions.

This module is divided into two parts: the opponent modeling function and the trust reasoning function. The opponent modeling function is what, using the information in the PR's knowledge

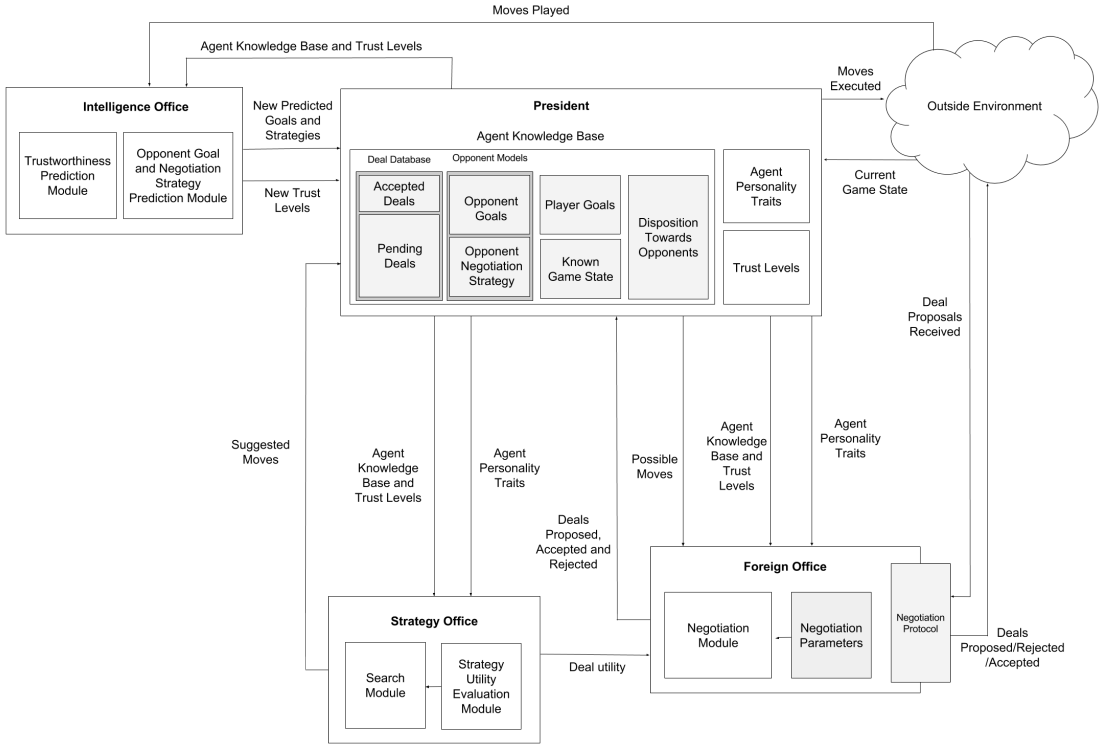

Figure 4.2: Detailed module structure of the Alpha architecture

base, outputs the predicted goals and their relative importance for each opponent, to be updated in the PR's knowledge base. How this is done is often specific to each game since the goals themselves as well as the actions and deals being analyzed are also specific to each game. The trust reasoning function is similar, outputting the trustworthiness values both for each opponent as well as for each individual deal, depending on how likely they are to be kept.

Like the FO, the IO can request the SO to calculate the utility of certain enemy moves and deals if necessary, using these values to aid in its predictions.

Different IOs can allow the developer to customize the opponent modeling and trust reasoning strategies, or lack thereof, of an agent. This module is especially useful in conjunction with the FO, since negotiations are the most likely to benefit from a good opponent model and accurate trust reasoning.

4.6 General Sequence of Play

In this section we will exemplify a typical play sequence of an agent built using the Alpha architecture.

At the start of the play the IO analyses any moves or deals proposed by other players and updates the trust levels per player and per deal. It also updates the predicted player goals and their negotiation strategies if possible.

Afterwards the PR can request the SO to find the best moves given the current situation in the game and other information in the PR's knowledge base such as any deals it entered into with other players. The SO searches the possibility space for the best moves and suggests a list of possible moves to the PR, with associated utility.

The PR can then ask the FO to negotiate any possible deals that could help in the execution of the suggested moves or completion of its goals. The FO exchanges messages with the other agents in the game and attempts to find deals that are likely to be accepted and fulfilled, considering opponent goals and trust levels, and at the same time increase the likeliness of success of the previously suggested moves or the player's goals. When a deal is found and proposed or accepted the FO updates the deal lists in the PR's knowledge base.

Finally, when the President feels confident to make a move, after possibly asking the SO for suggestions multiple times and after the negotiations of the FO are over, the PR chooses which move to execute. This choice is made depending on what confirmed deals exist in the PR's knowledge base and other personality traits such as its aggressiveness or its willingness to abide by deals. A PR with a high disregard for keeping its word may prefer to choose a higher utility move that conflicts with some deals while an honest PR may choose to always keep its deals and select the move to play accordingly. Some moves may depend entirely on the success of negotiations while others may be executed regardless, though their chances of success may increase if the FO is able to negotiate good deals.

After selecting the move to play the PR communicates its decision to the outside environment, such as a game server or other agents in the game.

4.7 Alpha Framework

In order to simplify the application of the Alpha architecture we developed a framework composed of several abstract classes which represent the modules and behavior described. These classes are shown in Figure 4.3 and define what each module should do as well as the data available and how this data is updated and communicated to and by each of the modules. Each module is defined in its own class and implements a specific interface. In addition to these there are several data classes that contain the data relative to the game being played and the agent itself, such as the knowledge base or the agent's personality traits.

In practice all a developer has to do to create an agent for a given domain is to implement the abstract classes of the modules he wishes to use. When implementing the modules the developer

must implement their abstract methods in order to define the domain specific negotiation strategies, protocols, heuristics, models and message handling. For the PR the developer defines the high level strategy for the agent, such as how it decides which goals are more important and what disposition it has towards other players. Optionally the developer can also specify actions for the agent to take before and after playing, such as initializing or cleaning up data. In the SO the developer implements the utility calculation functions for moves and deals, as well as the search strategy used to find and suggest moves to the PR. In the FO, the negotiation protocol and strategies are implemented as well as how the agent sends and receives the domain specific negotiation messages. Finally, in the IO are functions where the developer may implement trust reasoning and opponent modeling strategies.

The data produced by the different modules is automatically updated and available for use by all modules for their calculations so that if a developer decides to make use of the opponent goals predicted by the IO to enhance negotiations, it needs only access this data in the PR's knowledge base and use it when deciding which deals to propose and accept in the FO.

The developer must also implement the data classes with the domain specific definitions of what a move, a deal, a goal and an opponent are. After this is done the different modules can be attached to the PR and the agent be made to play the game.

4.8 Summary

Negotiation games have specific characteristics that encourage the use of strategies that can effectively deal with the social context of the game. The proposed Alpha architecture allows agents to be built in a generic and adaptable way, making use of negotiation, trust reasoning and opponent modeling to find good agreements that improve their chances of winning. The architecture is modular, so as to split the problem into more manageable sub-problems. It contains 4 main modules: the President, the Strategy Office, the Foreign Office and the Intelligence Office. These modules can interact among themselves and allow the developer to implement different trust reasoning and opponent modeling techniques as well as negotiation strategies, which all make use of, and contribute to, a central agent knowledge base stored in the President.

In the following chapter we introduce some of the agents created using the Alpha architecture, and more specifically taking advantage of the Alpha framework, for the games of Diplomacy and Werewolves of Miller's Hollow.

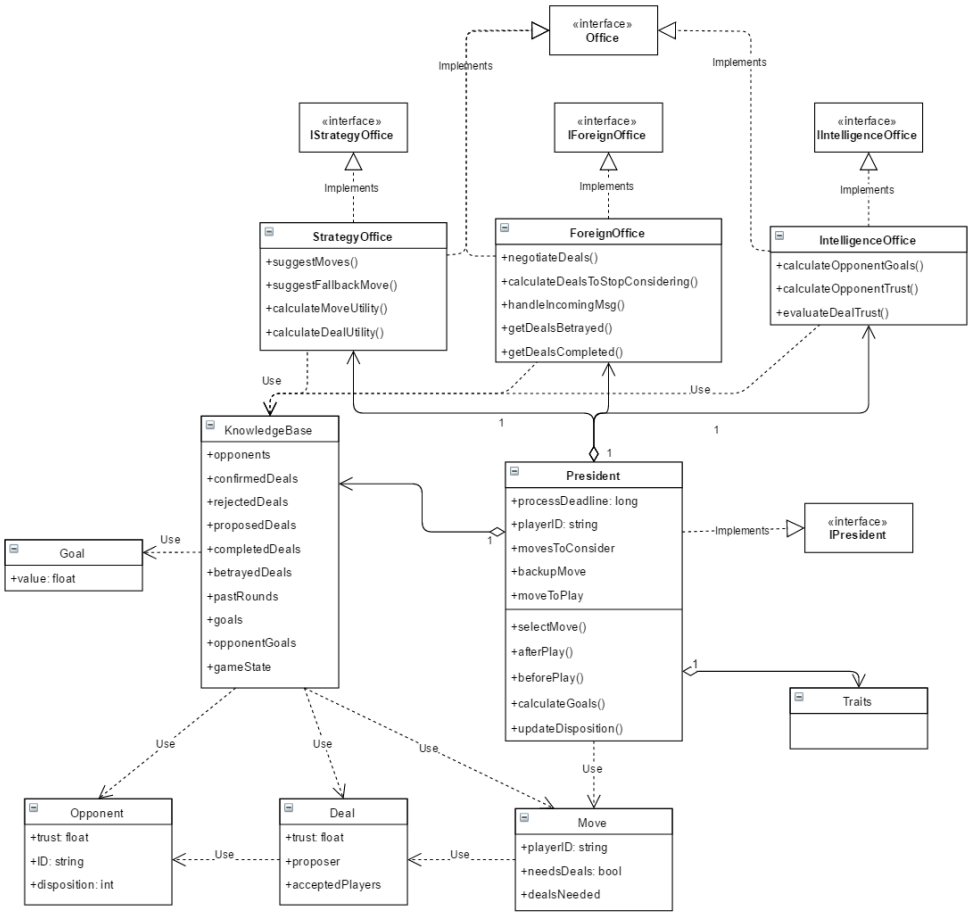

Figure 4.3: Class diagram of the Alpha framework along with the abstract methods that should be implemented for each module

Chapter 5

The Alpha Architecture in Practice

In this chapter we describe the details of the implementation of three different agents using the proposed Alpha architecture, for the games of Diplomacy and Werewolves of Miller's Hollow.

5.1 Diplomacy

As previously described in Chapter 2, one of the games selected to test the Alpha architecture was the game of Diplomacy. One of the reasons for choosing this game was the large amount of information, resources and agents available for it, as it has been a topic of interest for AI for a long time. For the development of the agents described in the following sections we made use of DipGame, one of the several available frameworks for the development of Diplomacy playing agents. DipGame provides a server to which our agents can connect to and submit their moves as well as communicate with other agents using the framework's communication language, the L Language. It also provides a set of classes and utilities with which to develop the agents, on top of which we applied the Alpha architecture.

We developed two Diplomacy playing agents which we will describe in this section. One is based on Dave de Jonge's D-Brane with several modifications and improvements, which we called AlphaDip. The other is a re-implementation of DipBlue, proposed by André Ferreira, following our architecture.

5.1.1 AlphaDip

AlphaDip is heavily based on D-Brane, the agent proposed by Dave de Jonge, using a modified version of its strategic module as well as the NB^3 algorithm described in Section 3.2 to search for the best moves. It has a few key improvements compared with D-Brane, the most notable ones being an improved strategic module, a defined strategy for negotiating coalitions and some ability to predict its opponents' goals and trustworthiness. Figure 5.1 shows a high level structure of how AlphaDip is implemented and how it related to the Alpha architecture.

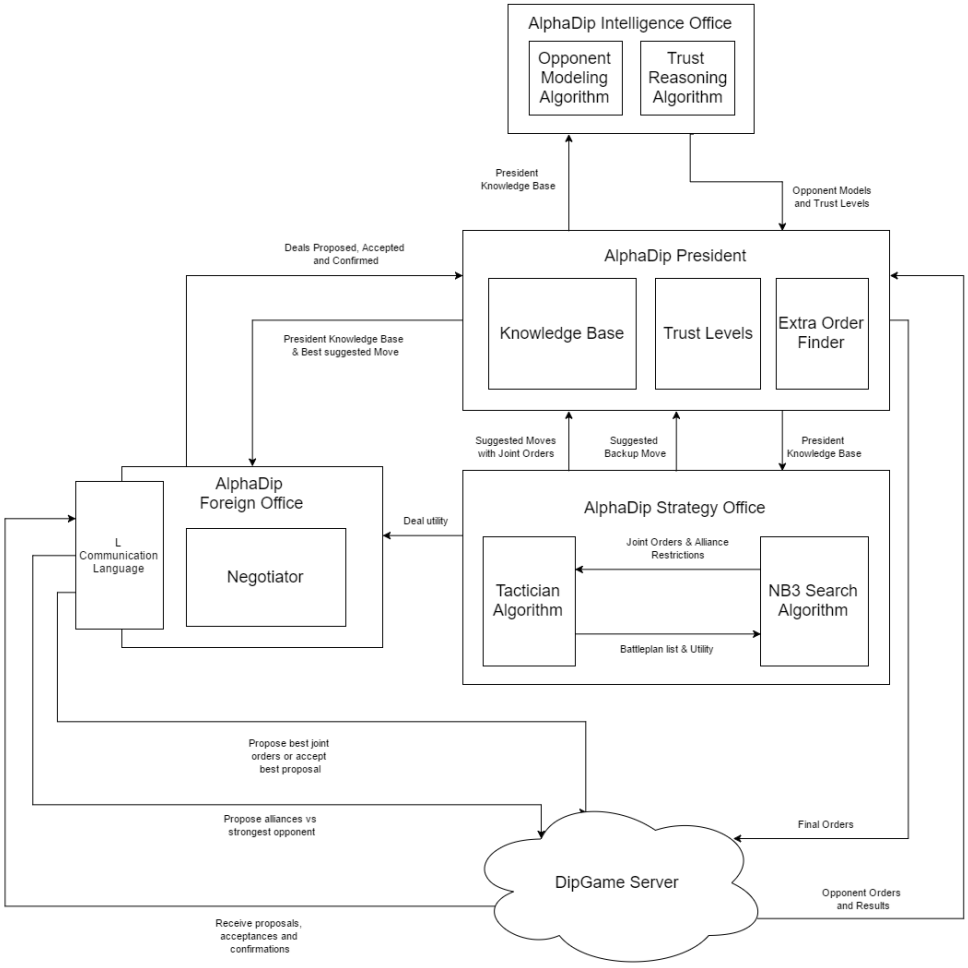

Figure 5.1: Structure of the Alpha architecture as implemented in AlphaDip

5.1.1.1 The President

AlphaDip's PR module is fairly straightforward, acting much the same way as described in the previous chapter. At the start of each round the PR asks the IO to analyze the previous round's moves and update the predicted trust values and predicted opponent goals in the PR's knowledge base.

In the context of AlphaDip, a move is considered a set of orders, one for each unit the player controls, as well as possible supporting orders from other players that may be necessary. The representation of player goals is relatively simple. Since in Diplomacy, the goal of the game is to capture as many supply centers as possible to win the game, a player's goals are simply a measure of how much they want to control a certain supply center at the moment. This is represented by a positive real number where the value 0 means no intention to control a supply center, 1 means neutral intention to control a supply center and values above that meaning greater intentions of controlling a supply center. On the other hand,the trust values for players stored by the PR are positive real numbers that are inversely proportional to the trustworthiness of these players. This

means that 0 represents full trust in a player, 1 represents neutral trust and values above that meaning lower levels of trust in a player.

When a round starts the PR first asks the SO for a fallback move that is expected to work even without any supporting deals with other players or any kind of negotiation. This move will be used by the PR in the eventuality that all negotiations fail, or in the absence of a FO. Afterwards the PR periodically asks the SO, who continually searches for the best moves, for suggested moves to consider. These suggested moves may include order commitments by other players in order to be feasible and so the PR passes the current best move it is considering on to the FO for it to negotiate any possible support deals. A deal is either a set of commitments for moves for the current round, or a proposal of alliance against a certain player. Currently, AlphaDip is not able to negotiate move commitments for the following rounds as that would increase the complexity of the agent tremendously.

Finally, after a certain negotiation deadline has passed and depending on whether the FO managed to negotiate any supporting moves deemed necessary, the PR picks the best move suggested to it that has a chance to succeed. Before submitting the orders the PR sends the chosen move to an Order Finder module that looks for any units that for any reason were not yet assigned an order and attempts to find good extra orders for these units that do not clash with the remaining orders.

5.1.1.2 The Strategy Office

AlphaDip's SO tries to find moves that maximize the number of controlled supply centers by the player, and is based on D-Brane's strategic module and the NB^3 algorithm.

The objective of the game is to take control of as many supply centers as possible, and we know from experience playing Diplomacy that it is fairly simple to calculate what supply centers are guaranteed to be controlled by a certain player if he plays a certain move.

As such, one way of determining the utility for a move is simply the number of supply centers that are ensured to be controlled by a player when it plays that move. This is the method used by D-Brane to calculate the utility of a move. AlphaDip calculates utility in a similar way, but introduces trust reasoning and the prediction of opponent goals in order to attempt to obtain a more accurate value than D-Brane. While D-Brane attributes the same score of 1 to every supply center, AlphaDip uses the player goals to influence the utility score. As described above, player goals are represented by a set of numbers meaning the degree to which a player wants to control each supply center. Instead of each supply center having the same value of 1, these values are used to calculate the utility of a move. Additionally, if a move requires supporting orders from other players to succeed, their trust values are taken into account when determining the utility value. This is so that SO suggests moves that are likely to be easy for the FO to obtain any necessary supporting move commitments from other players.

Equation 5.1 shows how the SO determines the utility of a move:

$$Utility: \quad U_p(m) \quad = \quad \frac{\sum_{i=1}^{n} I_m(i) \times g_p(i)}{t_p(m)} \tag{5.1}$$

Where $U_p(m)$ is the utility value of move m for player p, n is the total number of supply centers, I_m is a function where for each supply center it returns 1 if that supply center is sure to be controlled after move m and 0 if not, g_p is the goal value that player p has or is assumed to have for each supply center, and t_p is the average trust that player p has on all other players involved in the move or 1 if no other players are involved.

In order to find the best moves for a given round, the SO is divided into two components, the Tactician and the Searcher.

The Tactician attempts to find the best set of orders for a given player having certain restrictions such as possible fixed orders from agreements and who the player's current allies are. It then calculates the utility of that move as described above. To do this it uses the same method as D-Brane's strategic module described in [dJ15]. This method, described in Algorithm 1, splits the current round into several smaller battles for individual supply centers, and for each supply center tries to find possible sets of orders that attempt to capture or defend that supply center, which we call a battle-plan.

As described above it is simple to calculate which supply centers are guaranteed to be captured or defended if a certain battle-plan is employed, by comparing it with every possible enemy battle-plan (lines 9 and 11 of algorithm 1). If a battle-plan ensures the capture or defense of a supply center, we say that it is an invincible battle-plan. We can also determine pairs of invincible battle-plans, that is, two battle-plans that if executed simultaneously guarantee that at least one of them succeeds (lines 16-19). This is because while the opponents may be able to counter one battle-plan, this may prevent them from using units required to counter another battle-plan. After the Tactician has determined what invincible battle-plans and pairs of battle-plans there are, it uses an And-Or search to find the largest set of compatible invincible battle-plans or invincible pairs of battle-plans, that are also compatible with the fixed set of orders provided to it (line 25).

The Searcher uses the NB^3 algorithm to look for joint moves with other players that can maximize the utility for all players involved and that are likely to be able to be negotiated by the FO. The NB^3 algorithm and how it chooses to search the tree is also described in [dJ15]. Each node in the search tree is a set of joint orders for a certain supply center, and the path from a node in the tree to the root of the tree contains the set of fixed orders that are sent, along with any orders agreed in the PR's list of accepted deals for this round, to the Tactician in order to find the best possible compatible orders for the remaining units and the subsequent utility value of that node.

This means that each node in the tree is a set of restrictions on what certain units will do in that round, both for the player and its opponents, that would have to be negotiated as move commitments by the FO with the other involved players. While due to the way that the NB^3 algorithm blends negotiation with search this could be considered as a part of the negotiation process, and

Algorithm 1 Tactician Algorithm

```
playerAndOpponentBattleplans ← calculateBattleplans(gameState, player, allies)
playerAgreements ← getAgreementsFromPR()
for all bp ∈ playerAndOpponentBattleplans do
    if !isCompatible(bp,playerAgreements) then
        Remove bp from playerAndOpponentBattleplans
    end if
end for
playerPlans ← getPlayerPlans(playerAndOpponentBattleplans)
opponentPlans ← getOpponentPlans(playerAndOpponentBattleplans)
for all bp ∈ playerPlans do
    if !hasDefeatingPlans(bp, opponentBattleplans) then
        Add bp to invinciblePlans
    else
        defeatingPlans ← getDefeatingPlans(bp, opponentBattleplans)
        for all bp2 ∈ playerPlans do
            if isCompatible(bp,bp2) then
                defeatingPlans2 ← getDefeatingPlans(bp2, opponentBattleplans)
                if !hasLegalCombinationOfPlans(defeatingPlans,defeatingPlans2) then
                    Add bp and bp2 as a new pair to invinciblePairs
                end if
            end if
        end for
    end if
end for
bestBattleplans ← getBestCombinationUsingAndOrSearch(invinciblePlans,invinciblePairs)
return bestBattleplans
```

thus under the purview of the FO, no negotiation happens in the SO. Considering moves that also include other players in the search is simply a way to consider an extra set of possible moves that might be useful for the agents, depending on whether the FO eventually manages to negotiate the commitments necessary for their executions. By looking for joint moves like this the SO can find strategies and moves that would not be possible to execute if a player was acting completely independently, but that also bring a benefit not just to itself but also to the other players involved.

This utility value of a node for each player is used in the NB^3 algorithm as the intermediate value for that player and that node. The upper bound value for the node is calculated in a similar way except it considers any supply center next to a player's unit as ensured. The rationale for this is that a player's best possible result for that node would be obtained if each of its units next to a supply center captured it. On the other hand, the lower bound is calculated by taking only into account the joint move orders for that node, and any supply centers that they ensure.

Since the SO uses the trust values and predicted opponent goals when calculating the utility of a node, the search of the NB^3 algorithm will be directed towards nodes with joint moves with players in whom AlphaDip trusts, and who are more likely to accept the conditions of the joint commitments.

After a certain search time has elapsed the SO suggests to the PR the best move found by the Searcher, meaning one that maximizes the player utility and whose joint moves are likely to be successfully negotiated by the FO, as well as the best move found which is not dependent on any joint moves.

5.1.1.3 The Foreign Office

AlphaDip's FO handles all negotiation with other players and it performs two types of negotiation: coalition establishment with other players and order commitments for the current round. This is an improvement on D-Brane, which did not have a strategy for the establishment of coalitions, instead assuming that all D-Branes simply formed a coalition against all other players in the game.

The strategy employed to negotiate coalitions is similar to the strategy used by DipBlue. Like DipBlue, at the start of the game AlphaDip proposes a peace agreement to every other player in the game. After that, during the rest of the game the FO attempts to propose alliances against the stronger player in the game with which it is not in peace with. If a player's trust value rises above a certain threshold (meaning the player is less trusted) the peace with that player is broken. Conversely, if the trust value drops below a certain level (meaning the player is trusted) AlphaDip proposes peace to this player. Additionally, if the game has 4 or less players remaining AlphaDip immediately breaks any alliances it has with a player if that player controls 14 or more supply centers. This is so that AlphaDip does not let a player get too close to winning in the final stages of the game.

The FO also attempts to negotiate joint order commitments for the current round. The PR periodically asks the FO to negotiate deals concerning the moves being currently considered by the PR. Each time this happens, the FO compares the utility of the suggested moves with the utility of each of the proposals it received and either accepts the best proposal received if it has

more utility, or proposes any necessary joint order commitments required by the moves proposed by the PR.

If the FO proposes any deal for joint orders, it awaits for confirmation from all players involved in the deal before updating the PR's deal database and confirming to everyone involved that the deal has been accepted by everyone. If it accepted a proposal from another player, the FO awaits for the confirmation from that player that everyone has accepted before adding the deal to the confirmed deals in the PR's database.

If the FO receives a proposal from another player and it is compatible with any deals it has already accepted, it asks the SO to calculate the utility of that deal, and stores it in the proposed deals list of the PR's knowledge base for later consideration. The reason it does not choose to immediately accept or reject the proposal, as explained in [dJ15], is so that the SO is given some time to continue searching and looking for any possibly better options for other joint moves, before the agent commits to the proposed orders. By committing to an offer and adding it to the PR's confirmed deals list, the search performed by the SO is automatically constrained to only look for moves that satisfy the conditions in the accepted deals.

5.1.1.4 The Intelligence Office

The IO has the purpose of calculating the trust values for players and predicting their current goals in the game.

In order to update the trust values, the IO uses a strategy similar to DipBlue [dCF14], where trust in players increases steadily over the course of the game if no aggressive actions are taken by these players and decreases when aggressive actions are taken. The amounts by which the trust score increases and decreases are described in Algorithm 2 and are dependent on the current trust value the players have, as well as whether AlphaDip considers himself to be at peace or at war with these players.

As the trust value stored by the PR is inversely proportional to the trustworthiness of a player, this means that each round this value decreases as the trustworthiness increases (lines 5 and 8 in algorithm 2). Additionally, for every aggressive or negative action that AlphaDip encounters, such as attacks against itself or betrayals, the trust value for that player increases (lines 18 and 20), signifying that the trustworthiness of that player decreases.

Depending on the current trust value and disposition towards the player the way that its trust value changes is different, with this change being bigger on opponents with a higher trustworthiness (meaning a small trust value) and with which AlphaDip is at peace. This is accomplished with the use of an opponent ratio that is calculated for each player (lines 6, 10 and 12) and used when increasing that player's computed trust value (lines 17-18).

This way, if a player is highly trusted or in peace with AlphaDip, any aggressive actions it takes will have a bigger impact on that player's trust. On the other hand, if a player is not trusted or is at war with AlphaDip, any aggressive actions it takes have a smaller impact on that player's trust, since AlphaDip already expects that player to take aggressive actions.

In this way, a high emphasis is placed on betrayals from trusted or allied players, so that AlphaDip can quickly handle changing alliances during the games.

Algorithm 2 AlphaDip IO Trust Calculation Algorithm

```
1:  playerOpponents ← getOpponentsFromPR()
2:  opponentTrustValues ← getOpponentTrustFromPR()
3:  for all op ∈ playerOpponents do
4:    if isInPeace(op) then
5:      opponentTrustValues[op] ← opponentTrustValues[op] × 0.95
6:      opponentRatio ← opponentTrustValues[op] × peaceRatio
7:    else
8:      opponentTrustValues[op] ← opponentTrustValues[op] × 0.99
9:      if isInWar(op) then
10:       opponentRatio ← opponentTrustValues[op] × warRatio
11:     else
12:       opponentRatio ← opponentTrustValues[op]
13:     end if
14:   end if
15:   opponentHostileOrdersAndBetrayals ← getOpponentHostileOrdersAndBetrayals()
16:   for all ho ∈ opponentHostileOrdersAndBetrayals do
17:     if opponentRatio < 1 then
18:       opponentTrustValues[op] ← opponentTrustValues[op] + 0.1/opponentRatio
19:     else
20:       opponentTrustValues[op] ← opponentTrustValues[op] + 0.01
21:     end if
22:   end for
23: end for
24: return opponentTrustValues
```

The IO also attempts to predict its opponent's goals, that is, which supply centers it believes each player wants to control more, using a simple strategy exemplified in Algorithm 3. Each time a player takes an offensive action against a certain supply center, the IO increases the likelihood that that player wants to control that supply center (line 9 in algorithm 3). If a player takes no offensive actions against a supply center, or takes actions that would help another player capture that supply center such as support orders, the IO decreases the likelihood that the player wants to control that supply center (lines 5 and 12).

Like the trust values, these increases and decreases are dependent on the current values for each supply center. That way if a player is already expected to want control of a certain supply center any actions it takes have a small impact on the value for that supply center. On the other hand, if a player suddenly makes a move on a supply center that AlphaDip believed that player was not interested in, the value for that supply center will be affected more significantly. Figure 5.2 shows the update functions for the trust values and goal values when a player takes an aggressive action against a player or a supply center, respectively.

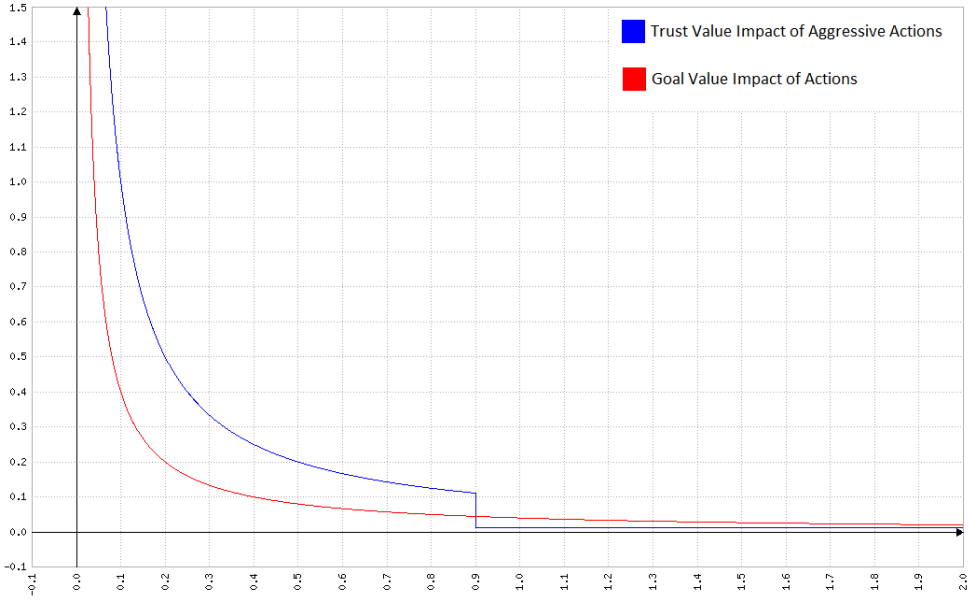

Figure 5.2: Amounts by which trust and goal values are incremented with each aggressive action or supply center attack respectively

5.1.2 DipBlue

In addition to AlphaDip, we re-implemented DipBlue in light of the Alpha architecture. Our version of DipBlue works exactly the same as the original DipBlue described in [dCF14]. In order to do this we split the DipBlue architecture into SO, FO and IO.

DipBlue's advisers are part of the SO, serving to calculate the utility value for the possible moves in the same way as originally. Unlike with AlphaDip, for DipBlue each move is a single order for a unit, and each order has a utility value assigned to it by the advisers. After finding the best orders, the SO suggests them to the PR, who asks the FO to negotiate any deals it thinks are necessary. The FO implements DipBlue's negotiation strategy, requesting supports from its allies for any moves that could use them and negotiating alliances and peace deals.

The execution of DipBlue's moves is not dependent on the success of negotiations with other players, though any supports negotiated may help with the success of those moves. As a result the PR will always execute the moves suggested to it by the SO, regardless of the result of any negotiations the FO attempts.

The IO performs calculations to update the trust values of the opponents in the same way that DipBlue did this. Unlike AlphaDip, DipBlue's IO does not make any attempt to predict its opponents' goals, as DipBlue did not have this ability.

Figure 5.3 shows a simple diagram exemplifying how the DipBlue architecture was mapped into the Alpha architecture.

Algorithm 3 AlphaDip IO Goal Prediction Algorithm

```
1: playerOpponents ← getOpponentsFromPR()
2: opponentGoals ← getOpponentGoalsFromPR()
3: for all op ∈ playerOpponents do
4:    for all g ∈ opponentGoals[op] do
5:       g ← g × 0.99
6:       ordersAndDealsSupportingGoal ← getOrdersAndDealsSupportingGoal(op,g)
7:       ordersAndDealsAgainstGoal ← getOrdersAndDealsAgainstGoal(op,g)
8:       for all o ∈ ordersAndDealsSupportingGoal do
9:          g ← g + 0.04/g
10:      end for
11:      for all o ∈ ordersAndDealsAgainstGoal do
12:         g ← g × 0.95
13:      end for
14:   end for
15: end for
16: return opponentGoals
```

5.2 Werewolves of Miller's Hollow

Werewolves of Miller's Hollow was the second game chosen to test the implementation of the Alpha architecture, for the reasons previously described in Chapter 2. As there were no available frameworks for the development of Werewolves of Miller's Hollow agents, we have implemented our own server using the Jade multi-agent framework. We then implemented an agent capable of communicating with this server and playing the game, which we nicknamed AlphaWolf. Figure 5.4 shows a high level description of how the Alpha architecture was applied in AlphaWolf.

In order to simplify the implementation, and because certain roles are more suited to be played physically with humans, we use a simplified version of the game with a subset of the available player roles and abilities.

In our version of the game there are 4 possible roles for the players: werewolves, villagers, seers and doctors. Werewolves have the goal of killing every other non-werewolf player in the game while every other player has the goal of killing the werewolves. The werewolves, seers and doctors each have a special ability that they can secretly perform during the night phase of the game. Werewolves can collectively vote on an enemy player to kill during the night. The seers can choose any player to investigate during the night, learning its secret role. Finally, the doctors are able to choose a player, who if attacked by the werewolves during the night will be healed and remain in the game, informing the doctor that this happened.

5.2.1 The President

The AlphaWolf's PR works much the same as described in Chapter 4. At the start of each phase of the game it requests the IO to update the other players' trust values and predicted goals. In the context of this game a player's goal is tied to its role and as such predicting a player's goal means

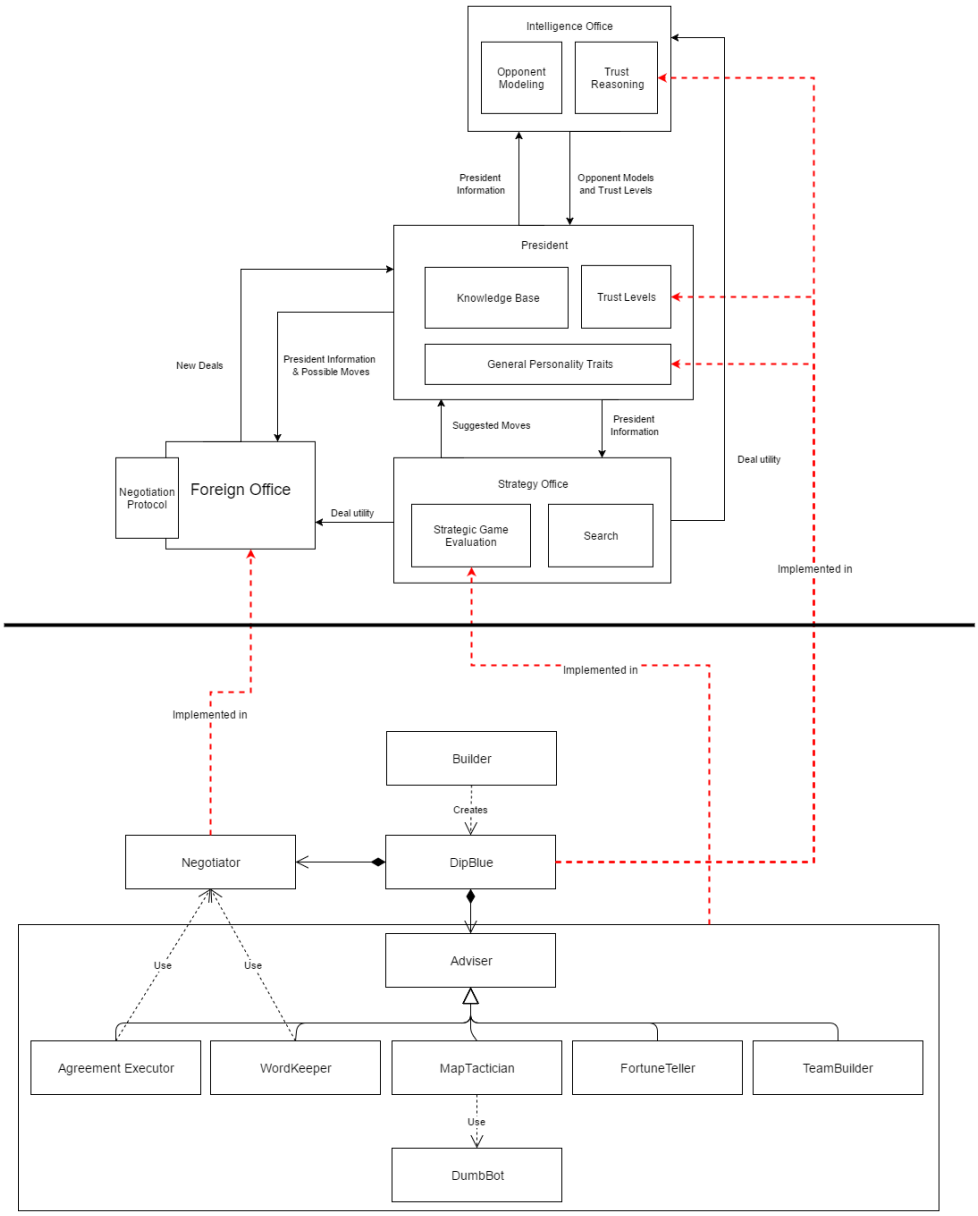

Figure 5.3: Mapping from the DipBlue architecture to the proposed generic architecture

predicting the likeliness that a player has a certain role. In the PR's knowledge base this means that each player is attributed a role certainty for each role, from 0 to 100%. The sum of all role certainties totals 100%, so that if a certain role has a certainty of 100%, the PR knows that player's role and consequently, its goals. A player's trust is represented by a positive real number, where 1 means neutral trustworthiness, 0 means no trustworthiness and values above 1 mean progressively higher trustworthiness.

Afterwards the PR requests the SO to suggest a good move. What constitutes a move in this context is dependent on the current phase of the game but it always involves choosing a player, to either vote out of the game or as a target for the player's ability during the night phase. After

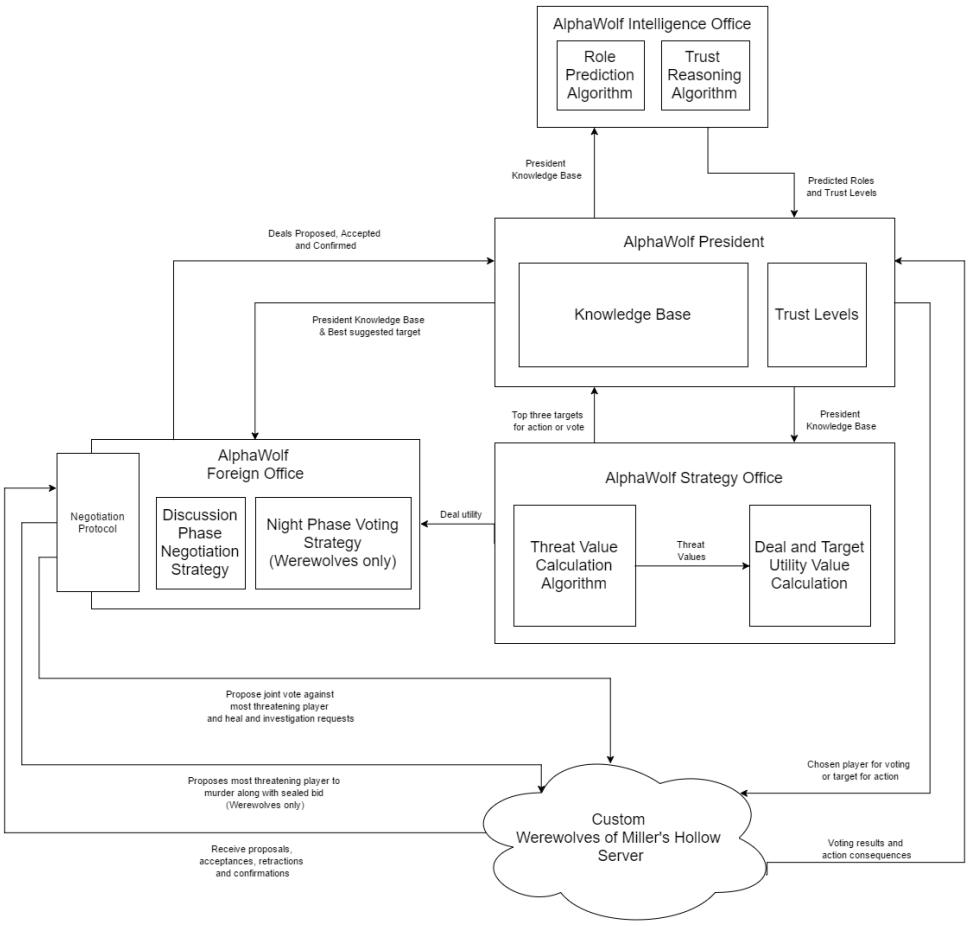

Figure 5.4: Structure of the Alpha architecture as implemented for AlphaWolf

a player is suggested by the SO, and depending on whether the current phase of the game allows negotiation between the players, the PR may ask the FO to attempt to negotiate with the other players for joint votes against some player or requests for investigation or healing. When the FO has finished negotiating, the PR decides to either vote or target a certain player for its special action. If no deal has been reached the player is randomly chosen from the list of players suggested by the SO, with players with higher utility being more likely to be picked, otherwise if a deal has been reached for a different player, the PR will take the action it agreed to on the deal.

5.2.2 The Strategy Office

The SO implements a simple strategy to suggest good potential players to either attempt to eliminate or protect, depending on the current phase of the game and the player's role and goals. The way it does this is by attributing to each player a threat score, which is a measure of what roles and goals the player believes an opponent to have (calculated by the IO) and how dangerous these roles are to the player.

In general terms, if a player is on the werewolf faction, roles that have the ability to gather more information or use abilities that hinder the werewolves' actions will have a higher threat level to the player. In the same way, if a player is on the villager faction (meaning that it's either a standard villager, a seer or a doctor), roles that have the ability to gather more information or hinder the werewolves are less likely try to kill the player, and thus are less threatening. The final threat value for an opponent is a weighted average of each role's base threat level for the player and how much he believes that the opponent has that role. Then depending on the current phase of the game and whether the actions available to the player will hinder an opponent, such as voting to kill it, or help another player, such as healing it, either this threat score is used as the utility for the move or its inverse is used, respectively.

Equation 5.2 shows how the threat value of a player is calculated:

$$ThreatValue: \quad T_p = \frac{\sum_{i=1}^{n} B_i \times C_i}{\sum_{i=1}^{n} B_i} \tag{5.2}$$

Where T_p is player p's threat value, n is the number of different possible roles a player can be, B_i is the base threat value for role i and C_i is the current certainty that player p has role i.

After the utility value of voting for or applying the special ability to all players is calculated, the top 3 results are selected and suggested to the PR.

The SO also has the purpose of calculating the utility of deal proposals. This utility is based on the previously mentioned threat value of the proposer of the deal, the threat value of the player whom the deal concerns, what type of action the deal is proposing and the trust of the player in the proposer of the deal. This calculation is described in Algorithm 4

The type of action proposed and the threat values for the proposer and the player affected by the proposal are used to calculate two values, one for the proposer and one for the target of the proposed action, representing how much the player is willing to help the proposer and hurt the target. For the proposer the inverse of his threat value is used for this (line 3 in algorithm 4), while for the target of the proposed action either the threat value or its inverse is used depending on whether the action is negative or positive for that player (lines 5-6).

These two values are then multiplied together with the trust on the proposer, representing how much the player trusts the proposer to abide by the deal and not take any actions against him, to reach the final utility value for the deal (line 9).

5.2.3 The Foreign Office

In Werewolves of Miller's Hollow the players can only communicate during certain phases of the game, namely the discussion phase and, for werewolves, the night phase. As the game is very reliant on communication between players negotiation is very important in order to obtain effective

Algorithm 4 AlphaWolf SO Deal Utility Calculation

```
1: target ← getTargetFromDeal()
2: proposer ← getProposerFromDeal()
3: proposerValue ← 1 / getThreatValue(proposer)
4: if dealActionIsPositive() then
5:     targetValue ← 1 / getThreatValue(target)
6: else
7:     targetValue ← getThreatValue(target)
8: end if
9: return targetVales × proposerValue × getTrustFromPR(proposer)
```

players. Otherwise players would not be able to coordinate their votes or use their abilities during the night. This is the purpose of the FO.

During the phases in which communication is allowed the PR may ask the FO to negotiate good deals that will lead to votes against a certain player, pleas for investigation of a player or for healing. The negotiation strategy that the FO uses for this is described in Algorithm 5. In the case of werewolves the PR will also ask the FO to negotiate with other werewolves who they will vote to eliminate during the night.

During the discussion phase (lines 1-5 in algorithm 5) if the player has not proposed any deals yet it will propose a joint vote against the player with the highest threat value to him suggested by the SO. It may also propose deals to investigate a player, or request a heal from any doctors available if its trust value in a player is too low.

When the FO receives a proposal for a deal from another player it asks the SO to calculate its utility and stores it in the proposed deals list in the PR's knowledge base (lines 8-11). After proposing its deals, the FO will periodically check the deals that were proposed to him and decide whether to accept one and retract any incompatible proposals it made, or keep waiting to see if other players accept its proposals (lines 22-34). It does this by comparing the utility value of each deal it received with its concession value, which is equal to the utility of the deal it proposed multiplied by a concession factor that decreases the longer the negotiation goes (line 21). If one or more deals proposed to it have higher utility than the concession value it accepts the one with the highest utility (line 29) and retracts its proposal unless it has already entered into a conflicting agreement with another player.

If the accepted proposal is a joint vote proposal the FO sends a confirmation message back to the proposer to inform him of its acceptance. The reason it does not do this for other kinds of proposals is that doing so would reveal the player's role to everyone, including the werewolves.

If another player accepts one of its proposals the FO locks this proposal and adds it to the confirmed deals list, making sure that it does not enter into any conflicting deals afterwards (lines 13-16).

In the case of the negotiation phase for werewolves coordinating during the night to vote for a victim, the strategy employed by the FO can be much simpler, since the werewolves have complete information about who the other werewolves are and can thus assume that they are

Algorithm 5 AlphaWolf FO Negotiation Strategy

```
1: if !hasProposed() then
2:    jointVoteProposal ← getHighestThreat()
3:    healAndInvestigateProposals ← getHealAndInvestigateProposals()
4:    myProposals ← jointVoteProposal ∪ healAndInvestigateProposals
5:    propose(myProposals)
6: end if
7: while !negotiationOver() do
8:    for all prop ∈ receivedProposals do
9:       utility ← getDealUtilityFromSO(prop)
10:      Store prop and its utility in the PR's list of proposed deals
11:      Remove prop from receivedProposals
12:   end for
13:   for all accept ∈ receivedAccepts do
14:      proposalAccepted ← getAcceptedProposal(accept)
15:      Lock proposalAccepted in the PR's list of confirmed deals
16:      Remove accept from receivedAccepts
17:   end for
18:   if timeSinceLastEvaluation > threshold then
19:      for all myProposal ∈ myProposals do
20:         if !isLocked(myProposal) then
21:            concessionValue ← getUtility(myProposal) × concessionFactor
22:            for all oppProposal ∈ getProposedDealsFromPR() do
23:               proposalUtility ← getUtility(oppProposal)
24:               if proposalUtility > concessionValue then
25:                  possibleProposalsToAccept ← possibleProposalsToAccept ∪ oppProposal
26:               end if
27:            end for
28:            if !isEmpty(possibleProposalsToAccept) then
29:               bestProposal ← getHighestUtilityProposal(possibleProposalsToAccept)
30:               accept(bestProposal)
31:               confirmedDeals ← confirmedDeals ∪ bestProposal
32:               retract(myProposal)
33:            else
34:               Keep waiting
35:            end if
36:         end if
37:      end for
38:   end if
39: end while
40: return confirmedDeals
```

working towards the same goals as themselves. In this case consensus between the agents is reached by means of a sealed bid mechanism, where each werewolf proposes one player to vote for as well as their preference level for that player. After all werewolves made their proposals, the bids are counted and the player with the highest preference level among all werewolves wins, with every werewolf voting for that player.

5.2.4 The Intelligence Office

AlphaWolf's IO has the function of attempting to predict a player's goals and its trustworthiness. As described previously, since a player's goals are tied to its role in the game predicting its goals is a matter of predicting its role.

In order to predict an opponent's role the IO analyses the actions that the player took in the previous rounds of the game, both its proposals and votes. The predicted role certainties for that opponent are thus a function of the threat values of the players that that opponent voted against, or proposed votes against, and the rounds in which that player took those actions. Algorithm 6 describes the calculation for the prediction of opponent goals by the IO.

The IO searches through each player's past actions and for each vote or proposal that that player made it calculates a vote or proposal damage value (line 9 in algorithm 6). This value indicates the likeliness that an action was taken with the intent of damaging the player's faction and is based on the threat values of the players who are the targets of that opponent's actions. A high threat value for the target of the action indicates that the action was not very damaging to AlphaWolf's faction, and may have even been helpful, and a low threat value indicates a damaging action, as that opponent was voting against players that are considered likely to be allies. That threat value is multiplied by a factor indicating how long ago that action was taken, so that actions that were taken long ago have a smaller damage value. This is because at the start of the game players have less information about each-other and are likely to take wrong decisions. As the game goes on and players get more information, votes are less likely to be mistakes and more likely to be intentionally damaging a certain faction.

For each action its vote damage is then used to calculate a scaling factor for each possible opponent role (line 10) and that scaling factor is finally used to scale the role certainties of each role proportionally (lines 11-12). The reason each role has a different scaling factor is because certain roles are more likely to have more information than others. So seers have a higher scaling factor than doctors, and doctors have a higher scaling factor than villagers.

In this way, for a player on the faction of the village, a low damaging or helpful action by an opponent increases the likeliness that that player is a seer more than that it is a villager. Likewise, if that opponent takes a damaging action, the likeliness that it is a seer diminishes comparatively more than the likeliness that it is a villager, since a seer would be less likely to commit damaging actions against the villagers, having more information about the player roles. Similarly for werewolves, if an opponent takes a high damaging action against them, it is more likely to be a seer than a villager, and if it takes a low damaging or helpful action for them, it is less likely to be a seer than a villager.

These scaling factors are then multiplied with each current role certainty, and the values for the roles are then re-scaled back so that they total 100% (line 14).

To calculate the trust values the IO analyses the previous round and checks for each opponent if it kept any agreements it accepted or if it voted against the player. If an opponent broke an agreement or voted against the player its trust value is decreased by a certain amount, otherwise its trust value increases.

Algorithm 6 AlphaWolf IO Role Prediction Calculation

```
 1: pastRounds ← getPastRoundsFromPR()
 2: opponents ← getOpponentsFromPR()
 3: opponentGoals ← getOpponentGoalsFromPR()
 4: for all op ∈ opponents do
 5:    for all round ∈ pastRounds do
 6:       roundAgeFactor ← getAgeFactor(round)
 7:       for all actions ∈ getOpponentActions(op, round) do
 8:          target ← getTarget(action)
 9:          voteDamage ← (getThreatValue(target) - getAverageThreatValue()) × roundAgeFac-
             tor
10:          scalingFactors ← calculateRoleScalingFactors(voteDamage)
11:          for all r ∈ opponentGoals[op] do
12:             r ← r× scalingFactors[r]
13:          end for
14:          normalizeOpponentGoals(opponentGoals[op])
15:       end for
16:    end for
17: end for
18: return opponentGoals
```

5.3 Summary

We developed three different agents using the Alpha architecture in different environments, two Diplomacy agents and one Werewolves of Miller's Hollow agent. All of these agents contain the same base modules and module interactions defined by the high level architecture and framework, with different specific implementations for opponent modeling and trust reasoning depending on the game and the agent.

AlphaDip is a Diplomacy playing agent based on D-Brane and DipBlue. It uses similar methods to D-Brane's strategic module with the addition of trust reasoning and opponent modeling. Unlike D-Brane it also has a defined negotiation strategy for the establishment of coalitions, based on the strategy used by DipBlue. AlphaWolf is a Werewolves of Miller's Hollow playing agent that is able to negotiate for joint votes or the use of different abilities on certain players, and uses opponent modeling to attempt to predict its opponents' roles in the game.

In the following chapter the results of the experiments and tests obtained with these agents will be presented and discussed.

Chapter 6

Tests and Experiments

This chapter describes the methodology and results of the tests and experiments performed with the agents implemented using the Alpha architecture. Afterwards an evaluation and discussion of the results is included.

6.1 Diplomacy

In this section we compare AlphaDip with two previously developed Diplomacy playing agents, DumbBot and DipBlue. DumbBot is a no-press agent usually used as a basis to test the relative effectiveness of different agents. DipBlue is an agent with negotiation and trust modeling capabilities which we use to test how well our agent performs in environments with other negotiating agents.

In order to compare our results we performed some of the same tests reported in [dCF14] and [dJ15] and compared our results. It should be noted that unlike the tests performed using D-Brane in [dJ15], which assumed that the D-Branes always formed a coalition against every other agent in the game, we allow our agents to negotiate at will, establishing and breaking coalitions depending on how the game is going.

In each of the experiments, we tested AlphaDip using 3 distinct configurations, in order to separately test the impact of negotiation and opponent and trust modeling in its performance. The configurations used were: an agent using only the PR and SO, an agent adding to this the FO and, finally, an agent using all 4 modules: PR, SO, FO and IO.

For every configuration, in each experiment we played a number of games of Diplomacy, stopping after the Winter of 1920 phase if the game had not finish by that phase. After a game had finished we ordered the players by ranking, from 1st to 7th, and collected the ranking results. Ranking is determined by the number of supply centers that a player controlled at the end of the game if that player was alive, or by the game phase in which that player was eliminated otherwise. Players with more supply centers or that were eliminated later than other players have a higher rank in the game and are thus considered better. If two or more players would get the same rank

Table 6.1: The average rank of 2 AlphaDips when playing with 5 DumbBots.

2xAlphaDips & 5xDumbBots	
AlphaDip Configuration	Average Rank
PR + SO	2.35 ± 0.15
PR + SO + FO	2.21 ± 0.21
PR + SO + FO + IO	2.11 ± 0.19

they are attributed the average of their ranks instead. As an example, if two players are tied for 1st place they get the average of rank 1 and 2, ending up with rank 1.5.

For configurations without the FO (and thus, with no negotiation capabilities) we played a total of 100 games. For configurations having the FO we set the negotiation deadline before the PR executes a move at 15 seconds per round, similar to the tests made in [dJ15] using D-Brane. Since these tests take considerably longer to execute we only played a total of 50 games per configuration in these cases.

All tests and experiments were performed in a laptop computer with 8GB of RAM and an Intel Core i5-6440HQ mobile CPU clocked at 3.5GHz.

6.1.1 AlphaDip Compared with DumbBot

Both in [dCF14] and [dJ15] the authors use the DumbBot to test the results of their agents, DipBlue and D-Brane respectively, by having two instances of their agents play against 5 instances of DumbBot. Similarly we performed those same tests, having two instances of AlphaDip play against 5 instances of DumbBot. Since we have two agents playing in each game, the best possible average rank for our agent in these tests is 1.5, while the worst possible average rank is 6.5. A player equally strong to the DumbBot would obtain an average rank of 4, which is the sum of all of the possible ranks divided by the number of players.

By comparing the average ranks of AlphaDip with the average ranks of DipBlue and D-Brane we can determine how well each agent performs comparatively against the DumbBot. The best rank achieved by DipBlue in these tests is 3.57 while the best rank obtained by D-Brane is 2.35, as reported in [dCF14] and [dJ15].

Table 6.1 shows the average rank obtained by AlphaDip in each configuration and their standard deviations. The results show that AlphaDip obtains an average rank between 2.1 and 2.4 in all configurations. It is notable that AlphaDip already obtains results much better than DipBlue and DumbBot, and equal to the best results obtained by D-Brane, even when using only PR and SO (that is, without negotiation and trust and opponent modeling, relying only on its strategic component).

6.1.2 AlphaDip Compared with DipBlue

In order to complement the previous experiments we also tested AlphaDip in an environment with a higher number of negotiating agents, having 2 AlphaDips play with 2 DipBlues and 3 DumbBots.

Table 6.2: The average rank of 2 AlphaDips when playing with 2 DipBlues and 3 DumbBots.

Average Ranking		
AlphaDips' Configuration	AlphaDips	DipBlues
PR + SO	2.38 ± 0.16	5.03 ± 0.20
PR + SO + FO	3.56 ± 0.26	3.44 ± 0.32
PR + SO + FO + IO	2.3 ± 0.22	4.73 ± 0.29

The 2 DipBlues played with the standard adviser configuration described in [dCF14] in all tests.

Like in the previous tests, the best and worst possible average rankings for both types of agents in this case are 1.5 and 6.5 respectively. Comparing these average ranks between both players will give us a good idea of how well they play compared to each other in an environment where the negotiation has a bigger impact on the outcome of the game.

The results for this experiment are shown in Table 6.2. These results show that when the AlphaDips are playing without both the FO and the IO or with all modules running they get an average rank between 2.3 and 2.4. When they play with the FO but without the IO, meaning that they negotiate without making any attempt to predict their opponent's goals or trustworthiness, their average ranking decreases significantly to 3.56. The DipBlues obtain significantly worse results than the AlphaDips in all tests except for the ones in which the AlphaDips were playing with the FO but without the IO, where they obtain a similar average ranking to AlphaDip of 3.44.

Interestingly, when the AlphaDips are playing with all modules, not only do they obtain better results, but so do the DipBlues, who obtain an average rank of 4.73 compared with the average rank of 5.03 when playing against AlphaDips with no negotiation capabilities.

6.2 Werewolves of Miller's Hollow

In this section we present the results for AlphaWolf, the Werewolves of Miller's Hollow playing agent we described previously. Unfortunately, due to the lack of any comparable agents developed for this game, we were only able to test the relative performance of our agent with different configurations of active modules.

In order to test the effect of the different modules on the performance of the agents we opted to have the werewolves always playing with all modules running, and changed only the configurations of any villager agents in the different tests. This way we can easily see the effect that each module has on the performance of the villagers. We tested 3 different configurations for this agent, just like with AlphaDip: just the PR and the SO, all modules except the IO and all modules active.

In each test we had the agents play 100 games in a 10 player game where 2 of the players were werewolves, and the remaining 8 were from the villager faction, with 1 seer, 1 doctor and the remaining 6 being standard villagers. This ratio of werewolf players to villager players was chosen because it is the recommended ratio for werewolves to villagers in the official Werewolves of Miller's Hollow rules. Increasing the number of werewolves significantly would allow the werewolves to coordinate among themselves and eliminate the villagers too easily. We recorded

Table 6.3: The win percentages and average number of remaining villagers for a team of 8 villagers playing against 2 werewolves.

Villagers' Configuration	Average Ranking Win Percentage	Avg # of Villagers Alive
PR + SO	27%	3.56
PR + SO + FO	46%	3.20
PR + SO + FO + IO	73%	4.88

the win percentage for the villagers over those games as well as the average number of villager agents left alive at the end of the game when the villager faction won. The results obtained are shown in Table 6.3.

Results show that without the FO or the IO the villagers obtain a poor performance, winning only 27% of the games. When adding the FO and the IO the performance increases considerably, with the villagers obtaining a win percentage of 73% with all modules active. Additionally, the inclusion of the IO increases the average number of villagers remaining when this group wins the matches, meaning that they are able to find the identity of the werewolves faster, and thus eliminate them before too many villagers are killed. On the other hand, when playing with just the FO and SO, without the IO, the average number of villagers alive decreases.

6.3 Analysis of Experimental Results

In the case of AlphaDip, our experiments show that AlphaDip plays significantly better than the DumbBot and DipBlue, even without having negotiation, opponent modeling or trust modeling.

When playing against DumbBots it appears that the inclusion of the IO does not affect the results significantly. This can be explained because the AlphaDips tend to keep their agreements and avoid attacking allies unless they themselves are attacked or their ally is getting close to victory. As such, trust is not as important as it would be in an environment with agents likely to betray the AlphaDips. This is supported by the fact that when playing against the DipBlues, who are more likely to betray them, the AlphaDips obtain significantly worse results if they are negotiating without the IO running. This decrease in performance can be explained by the AlphaDips entering into agreements and coalitions with the DipBlues, who end up betraying them or not keeping those agreements putting the AlphaDips in disadvantageous positions.

However, the results also show that, apart from the previously described case where negotiating without IO obtains worse results than no negotiation, the inclusion of negotiation does not significantly affect the performance of the AlphaDips. When playing with the DipBlues the average rank of the AlphaDips stays between 2.3 and 2.4, both when playing with only the PR and the SO and when playing with all modules active. A statistical t-test performed using the data from these tests gives us a value of approximately 0.653, which shows that the difference observed is not statistically significant. When playing with the DumbBots the results are also fairly similar for all 3 configurations, though there is a larger difference between the performance of the AlphaDips

with just the SO and the AlphaDips with all modules active. A t-test performed on the data from these tests results in a value of 0.554 for the results of the tests with all modules and without the IO, and a value of 0.109 for the tests with all modules and with just the SO. While the difference in performance for the first case does not appear to be significant, the second value indicates that the difference in the second case appears to be statistically significant. Further testing would be required to confirm the significance of the difference in performance of the AlphaDips with just the SO active and the AlphaDips with all modules active.

Nevertheless, the results are quite close, with only a difference of 0.24 in the average rank of the AlphaDips. This is similar to the results obtained by De Jonge in [dJ15], where he concludes that even though the NB^3 algorithm manages to find good joint moves for the players when they exist, the impact of these in the overall result of the game is negligible. As such, negotiating joint moves only for the current round is not enough to significantly increase the performance of the players – in order to obtain better results one would have to attempt to negotiate further rounds ahead as well.

An interesting observation is that when playing with the DipBlues, while the AlphaDips do obtain a small increase in their average rank when all modules are running compared to having just the PR and SO, the DipBlues themselves are also benefited when playing with AlphaDips capable of negotiating. The DipBlues obtain a better average ranking of 4.73 when playing with the AlphaDips with all modules active, compared with an average ranking of 5.03 when playing with AlphaDips incapable of negotiation. These observations were also corroborated by De Jonge's observations in [dJ15], where he found that other agents could also benefit from the deals discovered by agents running the NB^3 search algorithm. Another possible explanation is that the increase in effectiveness comes from the negotiation of coalitions among the players capable of negotiation, meaning that they would fight less among themselves and giving them an advantage against the DumbBots.

Unlike with Diplomacy, the results for the AlphaWolf agents show that negotiation, trust modeling and opponent modeling have a significant impact on the effectiveness of the agents. With the inclusion of the FO and the IO the performance of the agents steadily increases from a 27% win ratio to a 46% win ratio, and finally a 73% win ratio with all modules active. The inclusion of the IO also significantly increases the number of villagers remaining alive in games where the villagers win. This means that with the inclusion of trust and opponent modeling the players are able to identify the werewolves much earlier in the game, allowing for quicker victories. A interesting observation is that the average number of remaining villagers at end of the games when AlphaWolf has the FO but no IO is actually smaller that when AlphaWolf only has the SO. It is possible that because a majority of votes is required to kill a player during the voting phase, without negotiation less players die because of a lack of coordination, resulting in less deaths per round. This may allow the villagers to get lucky and vote against the werewolves early enough in the game that most villagers are still alive. With negotiation active, the AlphaWolfs may be more efficient in coordinating and deciding on who to kill, but their lack of opponent modeling and trust reasoning allows the AlphaWolfs playing as werewolves to take advantage of them and

propose votes against other players on the villager faction, until the villagers may eventually vote the werewolves out of the game after losing more of their own teammates.

One possible explanation for the difference in the relative impact of negotiation, trust and opponent modeling in the game of Werewolves of Miller's Hollow compared with Diplomacy is that in the latter the agents already implicitly have an idea of their opponent's goals. Since in Diplomacy the rules of the game encourage players to capture supply centers, players can already play with the assumption that other players will try to maximize their number of supply centers over the course of the game. On the other hand, in the game of Werewolves of Miller's Hollow players have no way to know at the start of the game what their opponents will be trying to do, since werewolves and villagers have entirely opposing objectives. This means that players have no way to predict an opponent's utility function at the start of the game, and must analyze a player's actions in order to determine it.

Negotiations may also have a greater impact in Werewolves of Miller's Hollow because each player only has a single vote, which will not be able to affect the outcome of the round by itself. Without coordination and organizing joint votes, players have a hard time completing their goals. In Diplomacy players can have differing numbers of supply centers and units, which allows certain players to affect the outcome of the rounds more than others; this leads strong players to use their superior strength to obtain their objectives even without negotiation.

6.4 Summary

Both AlphaDip and AlphaWolf were tested in several different scenarios in order to compare the impact and effectiveness of the introduction of the different modules defined in the Alpha architecture. The results show that the introduction of negotiation in the game of Diplomacy had only a small impact on the performance of the agents, and the introduction of trust and opponent modeling capabilities with the IO only had a significant impact when playing against agents that are likely to betray AlphaDip, as is the case with DipBlue. The results also show that AlphaDip appears to play significantly better than D-Brane, with its strategic module obtaining the same performance as D-Brane with negotiation, and the AlphaDips with all modules obtaining an even better average rank of 2.11.

In the game of Werewolves of Miller's Hollow the results show a much greater impact with the introduction of the three modules, with the win percentage of the villager faction more than doubling when all modules are active compared when only the PR and SO are active. Not only that, but the introduction of the IO had a significant impact on the average number of villager players alive in the games the villagers did win, indicating that the opponent modeling allowed the agents to more easily predict their opponents' roles and eliminate the werewolves.

Chapter 7

Conclusions and Future Work

In this chapter we describe and summarize the work done as well as the contributions to the field and answer the questions proposed in Chapter 1. We also suggest some avenues for possible future improvements on this work.

7.1 Conclusions

In Chapter 1 we discussed the motivations and objectives for this work. These objectives were the study of what elements were necessary to create effective agents that could play negotiation games with a mix of competition and cooperation as well as the development of a generic architecture that included these elements and could be used to facilitate the development of effective agents. Due to several characteristics of this type of games, described in Chapter 2, traditional exhaustive search strategies are hard to apply because the agents need to be able to take into account not just the strategic position in the board but also the social context of the games. The architecture proposed needed to be generic enough to be applied to a wide variety of cooperative games. In order to test the adaptability of this architecture we proposed to instantiate it using two very different games: Diplomacy and Werewolves of Miller's Hollow.

After researching previous works in the field of negotiation, trust, opponent modeling and general multi-agent systems as well as agents already implemented for games in this category, we found that the use of negotiation could be enhanced with the inclusion of opponent modeling and trust modeling. We also found that due to the vast differences in the rules and characteristics of negotiation games, in order to be generic enough to be applicable to different games the architecture proposed would have to make few or no assumptions about the negotiation protocols and languages used as well as allow for the implementation of different negotiation, trust and opponent modeling strategies depending on the required capabilities for the agent.

Taking inspiration from both the modular architectures of DipBlue and the Israeli Diplomat we decided that the best way to match the criteria required for this architecture would be to separate the different necessary capabilities that the agents should have into 4 separate modules. The President serves as a coordinator and final decision maker for the agent, ordering the remaining

modules and requesting suggestions when necessary. The Strategy Office handles all of the game-specific strategic analysis and is responsible for suggesting actions for the President to take. The Foreign Office handles all interaction with other agents with the environment, being in charge of negotiating deals according to the wishes of the President. Lastly, the Intelligence Office is in charge of analyzing opponent actions in order to derive useful information about how they are likely to behave in the future, that can be used to help the other modules. This modular approach makes it simple to change the capabilities of an agent by making changes or improvements to a single module, or changing the module configuration depending on the situation.

Since several of the modules could conceptually have the need to use much of the same data and information available to the agent, we decided to aggregate all of the information the agent contains about the outside environment in the President's knowledge base, which can be accessed by the remaining modules to help with calculations or to update it with relevant new data. This knowledge base includes information regarding the state of the game, what actions were taken previously by other players in the game that the agent knows about, what are the agent's current goals, who the player's opponents are, what the current disposition towards those opponents is, how trustworthy they are, what goals they have and what strategies do they use, what deals have been proposed and confirmed and finally, how much trust there is in those deals.

We believe that the proposed Alpha architecture is generic enough to be applied to many different games. It is also modular enough to be simple for developers to alter the capabilities of the agents and obtain different agents or agent configurations. However, it also allows for different, game-specific, negotiation strategies and trust and opponent modeling strategies to be implemented depending on the characteristics of the specific game. We believe that this can be a helpful contribution to the field of multi-agent systems in the context of cooperative negotiation games.

To test it, we developed a simple Alpha framework that provides abstract classes and methods representing the different modules described in the architecture. By implementing these classes and defining the domain specific strategies, protocols and heuristics in the provided functions, an agent can be implemented to play several types of cooperative negotiation games.

We implemented three different agents using the Alpha architecture and framework, for the games of Diplomacy and Werewolves of Miller's Hollow. AlphaDip is an agent with strategies inspired by D-Brane and DipBlue, with the inclusion of opponent modeling to make predictions about an opponent's intention to capture certain supply centers, as well as a negotiation strategy for the establishment of coalitions, which was not present in D-Brane. DipBlue was also re-implemented using the Alpha architecture, showing a very different agent also running using the same general high level structure. AlphaWolf is a Werewolves of Miller's Hollow agent that also includes negotiation, trust and opponent modeling capabilities, allowing it to predict its opponents' roles and negotiate deals accordingly. We found that the Alpha architecture was relatively simple to be adapted to very different environments and types of games, and the structure easy to understand and work with for the different implementations.

In Chapter 6 we presented the results of the tests and experiments run using the developed

agents. These results show that AlphaDip was in general superior to both DipBlue and D-Brane, obtaining better average ranks in the games played. However, the inclusion of negotiation capabilities did not have a very significant impact on the performance of the agent. For this reason the inclusion of opponent modeling and trust modeling also did not have a very large impact except when AlphaDip attempts to negotiate with agents who are likely to betray him, in which case the inclusion of the Intelligence Office provided a significant increase in performance. These results are probably due to the specific strategies and heuristics used by AlphaDip's Strategy Office and President, that already find good moves for the player regardless of any negotiation, and the negotiation strategy used, that is only able to negotiate for joint moves in the current round of the game. An interesting observation was that with the inclusion of negotiation, while AlphaDip's performance remained similar, there was an increase in the performance of other negotiating agents in the game, namely DipBlue. This is supported by the results obtained by Dave de Jonge in [dJ15], where he found that the use of the NB^3 algorithm allowed the agents to find deals that benefited not only the agent itself, but also other agents involved in those deals.

The results obtained for AlphaWolf show that the inclusion of the Foreign Office and Intelligence Office, with their respective capabilities, had a larger impact in the performance of the agent. This indicates that negotiation, trust and opponent modeling are more important in Werewolves of Miller's Hollow than in Diplomacy. One reason for this may be that because agents have no information about their opponents' roles and faction at the start of the game, they have no way to predict their opponents' utility functions and goals. This makes it much harder to decide what are the best deals to propose and accept at the start of the game. In Diplomacy players are able to make some assumptions about their opponents' ultimate goals in the game because of how the rules encourage players to capture supply centers. Another important aspect of Werewolves of Miller's Hollow that makes negotiation important is that players always have the same power and influence in the outcomes of each round. This means that a single player is not able to alter the course of a round without the support from other players, while in Diplomacy a strong player can oftentimes overpower smaller players without the aid from anyone else in the game.

7.2 Future Work

In this section we discuss some possible improvements to the architecture proposed as well as to the agents developed.

While the developed architecture is very modular and allows agents to be built upon it and make use of negotiation, trust and opponent modeling, it can still be made more generic and adaptable, so that developers can more easily create new agents for different types of games.

While the architecture allows developers to define the different strategies for negotiation, trust reasoning and opponent modeling that they want to use depending on the specific game, such as DipBlue's trust reasoning strategy or D-Brane's NB^3 algorithm, this process can be made simpler by the inclusion of generic strategies that can be applied equally to any environment. The inclusion of a generic way to predict opponent goals and strategies, calculate trust values and decide what

deals to accept, based on the knowledge base of the President, would simplify the process of developing an efficient agent even more.

An even larger step in obtaining a truly generic system would be the inclusion of some form of abstract understanding about the rules of the game being played and the board state, which could be defined by the developer using a formal language such as the one used in the Zillions Of Games software. This language would have to be able to abstractly define the rules of the game, what deals are possible for the players to make and how communication between players can happen in the game. With this capability it would be possible to have a system that could generate agents able to play and negotiate in many different types of negotiation games, by simply providing it with a file containing this abstract description of the game. The system could then generate the appropriate strategies to be able to follow the rules and play the game effectively. This would be a significant improvement on the Zillions of Games software which does not allow for the creation of players able to negotiate among themselves.

Currently, in the implemented agents all modules always have the same impact on the decisions of the President. With the use of different traits in the President it could be made to give greater preference to some modules over others, allowing some modules to have a greater impact when altering the President's knowledge base. This could allow the President to, for example, ignore deals made by the Foreign Office more easily in certain cases, or ignore the predictions of the Intelligence Office if it was shown that they were often wrong. One could then use Genetic Algorithms to optimize these traits over time, allowing for the creation of agents that attribute the appropriate importance to different modules depending on their performance in the game.

The agents implemented during the course of this work, while generally efficient, could also be improved. Besides improvements to the heuristics and algorithms in AlphaDip's Strategy Office, one major improvement could be to allow the agent to search for and negotiate deals regarding movement commitments for several rounds instead of only the current round. This could possibly increase the impact of negotiation in the performance of the agent. Another possible improvement in the area of negotiation is to increase the types of deals that the agent is able to negotiate, such as information exchanges or the negotiation of demilitarized zones. Other communication capabilities could also be added, such as allowing the agent to express likes and dislikes about its opponent's actions or suspicions about other player's relations in the game. Finally, in terms of opponent modeling the strategy used could also be improved by predicting and taking into consideration a larger variety of possible objectives for its opponents, such as a player wanting to weaken a specific other player or wanting to control a specific non-supply center territory.

In the case of AlphaWolf and the Werewolves of Miller's Hollow server implemented, important improvements include adding the capability for new roles and new special abilities to the server beyond the ones already implemented.

One key and highly effective strategy when playing Werewolves of Miller's Hollow that human players take advantage of but that AlphaWolf does not, is the possibility of bluffing and making opponents believe one has a different role than its true role. This is especially useful for werewolf players, where if they can make villager players believe them to have a trustworthy role such as

a seer they can more easily convince them to vote for their own allies. However it is also a risky strategy that requires a good understanding of the game's social context and the ability to know when to lie and what information to reveal. In order to be able to bluff, AlphaWolf would require the ability to expressly lie about its role and abilities or to reveal false information to other players, which it currently does not possess. It would also need to be able to correctly determine when to execute this strategy depending on the current context of the game. If correctly implemented, this ability could make AlphaWolf much more effective, especially when playing with human opponents.

References

[AØ13] Einar Nour Afiouni and Leif Julian Øvrelid. Negotiation for strategic video games. Master's thesis, Norwegian University of Science and Technology, June 2013.

[Asm] Asmodee. Werewolves of millers hollow. Available at `https://www.asmodee.us/en/games/werewolves-of-millers-hollow/products/werewolves-of-millers-hollow/`. Accessed: 2016-02-10.

[BDL+99] M. Beer, M. D'inverno, M. Luck, C. Preist N. Jennings, and M. Schroeder. Negotiation in multi-agent systems. *The Knowledge Engineering Review*, pages 285–289, 1999.

[Bur11] C. Burnett. *Trust Assessment and Decision-Making in Dynamic Multi-Agent Systems*. PhD thesis, University of Aberdeen, 2011.

[Cal] Allan B. Calhamer. *The Rules of Diplomacy*. Avalon Hill, fourth edition.

[Civ] CivFanatics. Civ4 sdk - civilization modding wiki. Available at `http://modiki.civfanatics.com/index.php/Civ4_SDK`. Accessed: 2016-02-10.

[Cor] Zillions Development Corporation. Zillions of games – unlimited board games & puzzles. Available at `http://www.zillions-of-games.com/`. Accessed: 2016-02-10.

[Cor13] João Correia. Pgdl: Sistema para definição genérica de jogos de poker. Master's thesis, University of Porto, June 2013.

[DAI] DAIDE. Daide homepage. Available at `http://www.daide.org.uk/w/index.php?title=Main_Page`. Accessed: 2016-02-10.

[dCF14] André Filipe da Costa Ferreira. Dipblue: a diplomacy agent with strategic and trust reasoning. Master's thesis, Universidade do Porto, 2014.

[dJ13] Dave de Jonge. Negotiation algorithms for large agreement spaces. In Francesca Rossi, editor, *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 3209–3210. AAAI Press, 2013.

[dJ15] Dave de Jonge. *Negotiations over Large Agreement Spaces*. PhD thesis, Universitat Autonoma de Barcelona, 2015.

[Dro95] A. Drogoul. When ants play chess (or can strategies emerge from tactical behaviours?). *Reaction to Cognition — Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-93 (LNAI Volume 957)*, pages 13–27, 1995.

[FCR16] André Ferreira, Henrique Lopes Cardoso, and Luís Paulo Reis. Strategic negotiation and trust in diplomacy - the dipblue approach. In N.T. Nguyen et al., editor, *Transactions on Computational Collective Intelligence XX*, LNCS 9420, pages 179–200. Springer, 2016.

[FS09] Angela Fabregues and Carles Sierra. A testbed for multiagent systems technical report iiia-tr-2009-09. Technical report, IIIA-CSIC, 2009.

[FS11] Angela Fabregues and Carles Sierra. Dipgame: A challenging negotiation testbed. *Engineering Applications of Artificial Intelligence*, pages 1137–1146, October 2011.

[FSJ98] P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents. *Int. Journal of Robotics and Autonomous Systems*, pages 159–182, 1998.

[FSJB99] P. Faratin, C. Sierra, N.R. Jennings, and P. Buckle. Designing responsive and deliberative automated negotiators. In *Proceedings of the AAAI Workshop on Negotiation: Settling Conflicts and Identifying Opportunities*, pages 12–18. AAAI, 1999.

[FWJ06] Shaheen S. Fatima, Michael Wooldridge, and Nicholas R. Jennings. Multi-issue negotiation with deadlines. *Journal of Artificial Intelligence Research*, pages 381–417, 2006.

[Gib92] R. Gibbons. *A Primer in Game Theory*. Harvester Wheatsheaf, 1992.

[Har74] J. C. Harsanyi. An equilibrium-point interpretation of stable sets and a proposed alternative definition. *Management Science*, page 1472, September 1974.

[Hil] Avalon Hill. Diplomacy. Available at `http://avalonhill.wizards.com/games/diplomacy#submenu-page`. Accessed: 2016-02-10.

[HL95] Michael R. Hall and Daniel E. Loeb. Thoughts on programming a diplomat. *Heuristic Programming in Artificial Intelligence*, 1995.

[Hou04] C. Hou. Modelling agents behaviour in automated negotiation. Technical Report KMI-TR-144, Knowledge Media Institute, The open University, Milton Keynes, UK, May 2004.

[JAMSU11] Hamid Jazayeriy, Masrah Azmi-Murad, Nasir Sulaiman, and Nur Izura Udizir. The learning of an opponent's approximate preferences in bilateral automated negotiation. *Journal of theoretical and applied electronic commerce research*, pages 65–84, 2011.

[JH05] Stefan J. Johansson and Fredrik Håård. Tactical coordination in no-press diplomacy. *Proceedings of the fourth international joint conference on Autonomous agents and 4 multiagent systems - AAMAS '05*, page 423, 2005.

[JIB07] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, pages 618–644, March 2007. ISSN 0167-9236.

[JO05] Stefan Johansson and Fredrik Olsson. Mars — a multi-agent system playing risk. *Eighth Pacific Rim International Workshop on Multi-Agents (PRIMA)*, 2005.

[Jon10] Dave De Jonge. Optimizing a diplomacy bot using genetic algorithms. Master's thesis, UAB, 2010.

[KL95] S. Kraus and D. Lehmann. Designing and building a negotiating automated agent. *Computational Intelligence*, pages 132–171, 1995.

[LBS08] Kevin Leyton-Brown and Yoav Shoham. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. Morgan & Claypool Publishers, 2008.

[Leo10] Robert Leonard. *Von Neumann, Morgenstern, and the Creation of Game Theory*. Cambridge University Press, 2010.

[Mar94] S. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, 1994.

[MSJ98] N. Matos, C. Sierra, and N.R. Jennings. Determining successful negotiation strategies: an evolutionary approach. *Proceedings of 3rd Int. Conf. on Multi-Agent Systems (ICMAS-98)*, 1998.

[Nas51] John Nash. Non-cooperative games. *The Annals of Mathematics*, pages 286–295, September 1951.

[New02] Monty Newborn. *Deep Blue: An Artificial Intelligence Milestone*. Springer, 2002.

[Pru81] D.G. Pruitt. Negotiation behavior. *Academic Press New York*, 1981.

[PSM11] I. Pinyol and J. Sabater-Mir. Computational trust and reputation models for open multi-agent systems: a review. *Artificial Intelligence Review*, pages 1–25, 2011. ISSN 0269-2821.

[Rai82] H. Raiffa. *The art and science of negotiation*. Belknap Press, 1982.

[Ras06] Eric Rasmusen. *Games and Information: An Introduction to Game Theory, 4th Edition*. Wiley-Blackwell, 2006.

[Rib08] João Santos Ribeiro. Darkblade - um agente para diplomacia. Master's thesis, Universidade de Aveiro, 2008.

[Rit03] Alan Ritchie. Diplomacy - a.i. Master's thesis, University of Glasgow, September 2003.

[Rub82] Ariel Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica 50*, pages 97–109, 1982.

[RZ] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of encounter: designing conventions for automated negotiation among computers*.

[Sah06] S. Saha. Improving agreements in multi-issue negotiation. *Journal of Electronic Commerce Research*, 2006.

[SHM+16] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks & tree search. *Nature*, 2016.

[Sie11] Carles Sierra. Nb3: Negotiation-based branch & bound. Technical Report TR—IIIA—2011–03, IIIA-CSIC Campus de la Universitat Autònoma de Barcelona, 08193 Bellaterra, Barcelona, April 2011.

[Tho03] Robert S. Thomas. *Real-time Decision Making for Adversarial Environments Using a Plan-based Heuristic*. PhD thesis, Northwestern University, 2003.

[Urb13] M.J. Urbano. *A Situation-aware and Social Computational Trust Model*. PhD thesis, University of Porto, 2013.

[vKLH13] Thijs van Krimpen, Daphne Looije, and Siamak Hajizadeh. Hardheaded. *Complex Automated Negotiations: Theories, Models, and Software Competitions*, pages 223–227, 2013.